IDA

INSTITUTE FOR DEFENSE ANALYSES

A Review of Computer-Based Human Behavior Representations and Their Relation to Military Simulations

John E. Morrison

August 2003

IDA Paper P-3845
Log: H 04-000049

This work was conducted under contracts DASW01 98 C 0067/ DASW01 02 0012, Task AK-2-2190, for the Defense Modeling and Simulation Office. The publication of this IDA document does not indicate endorsement by the Department of Defense, nor should the contents be construed as reflecting the official position of that Agency.

INSTITUTE FOR DEFENSE ANALYSES

IDA Paper P-3845

A Review of Computer-Based Human Behavior Representations and Their Relation to Military Simulations

John E. Morrison

PREFACE

This paper was prepared as part of an IDA task entitled "DMSO Mission Review" sponsored by the Defense Modeling and Simulation Office (DMSO). The IDA task leader was Dr. Dexter Fletcher. Technical cognizance of the work was assigned to CAPT Michael Lillienthal, Director of DMSO.

CONTENTS

TABLES

EXECUTIVE SUMMARY

One of the goals of the Defense Modeling and Simulation Office (DMSO) has been to promote the development and assessment of computational human behavior representations (HBRs) that potentially provide synthetic forces—Red and Blue—for live, virtual, and constructive military simulations. The purpose of this paper is to review the domain of HBRs that could be integrated with military simulations. The intent is to provide the modeling and simulation (M&S) community an understanding of specific HBR models and to identify specific interoperability problems.

Table ES-1 presents the names and associated acronyms or abbreviations of the 19 HBR models reviewed in the present study. The following topic areas were chosen to frame the discussion of each model and to orient the review toward those who are interested in applying HBR models to military simulations:

- Model Purpose and History of Development
- Principal Metaphors and Assumptions
- Cognitive/Behavioral Functions Represented
- Applications
- Technical Considerations
- Evaluation.

To make generalizations across these models, each HBR was evaluated on whether it supported the following cognitive and behavioral functions:

- Perception
- Psychomotor performance
- Attention
- Situation awareness (SA)
- Short-term memory (STM)
- Long-term memory (LTM)
- Learning
- Decision-making
- Problem solving
- Cognitive workload
- Emotional behavior
- Social behavior.

Table ES-1. HBR Models Reviewed in the Present Study

Model Name	Acronym/ Abbreviation
Atomic Components of Thought	ACT
Adaptive Resonance Theory	ART
Architecture for Procedure Execution	APEX
Business Redesign Agent-Based Holistic Modeling System	Brahms
Cognition and Affect Project	CogAff
Cognition as a Network of Tasks	COGNET
Cognitive Complexity Theory	CCT
Cognitive Objects within a Graphical EnviroNmentT	COGENT
Concurrent Activation-Based Production System	CAPS
Construction-Integration Theory	C-I Theory
Distributed Cognition	DCOG
Executive Process/Interactive Control	EPIC
Human Operator Simulator	HOS
Man-machine Integrated Design and Analysis System	MIDAS
Micro Systems Analysis of Integrated Network of Tasks	Micro SAINT
Operator Model ARchitecture	OMAR
PSI	PSI
Situation Awareness Model for Pilot-in-the-Loop Evaluation	SAMPLE
State, Operator, And Result	Soar

The analysis suggested some generalizations concerning the current state of the art in human behavior modeling:

- Decision-making is a universal function of all models. This is not surprising since the overwhelming majority of HBR models can be classified as "rule-based" in their use of if-then, condition-action, "production" rules.
- All the models can represent either STM or LTM. Most of the models are derived from the tradition of human information processing, which employs the tools and the metaphors of computer science. Memory storage is a central concept to both disciplines.
- The "front-end" of cognition (perception and attention) is well represented in most models. Similarly, the ultimate output from cognition (psychomotor action) is modeled, at least to a minimal degree, in most models. The attention to inputs and outputs may also be a reflection of the information-processing tradition in human behavioral modeling.

- Learning and problem-solving functions are represented in only five models. Taken together with the ability of many models to emulate the "front-end" of cognition, this orientation suggests the following: Whereas most HBR models may be good at *reacting* to expected situations (i.e., situations for which they are programmed), they may not be so good at *adapting* to novel situations.
- The capability to emulate SA is explicitly represented in only 4 of the 19 models. However, this deficiency may be more apparent than real. Since most models have the capability to emulate perception and attention, they may also represent SA functions if appropriately modified.
- Very few of the models have the capability to simulate emotional or social behaviors. Recent publications and presentations suggest that these may be "growth areas" for current and future HBR models.

I. INTRODUCTION

A. BACKGROUND

The role of the Defense Modeling and Simulation Office (DMSO) has been to coordinate modeling and simulation (M&S) efforts within the Department of Defense (DoD). One of DMSO's specific goals is to promote the development, assessment, and reuse of computational human behavior representations (HBRs) that potentially provide synthetic forces—Red and Blue—for live, virtual, and constructive military simulations. The purpose of this paper is to review HBRs that could be integrated with military simulations. The intent is to provide the M&S community an understanding of specific HBR models and to identify specific interoperability problems.

The HBR models reviewed in this paper share a common intent: to provide comprehensive and integrated computational models of human behavior. However, among the models, marked differences exist with respect to focus and/or resolution. For instance, some of the models reviewed could be viewed as examples of "unified theories of cognition" (Newell, 1990). That is, the models comprise a single set of covert mental mechanisms that simulate the full range of human cognitive activities—from object perception and recognition to abstract problem solving. In contrast, another group of models may be identified as "performance models" because they simulate only the observable outcomes of covert and overt behavior. The distinction between cognitive and performance models, while conceptually valid, is that most models incorporate aspects of performance and cognition for practical reasons. For instance, cognitive models must have a performance component to simulate human actions that result from mental processing. Similarly, performance models often incorporate covert processing to simulate militarily important but essentially mental phenomena, such as decision-making and situation awareness (SA).

Another differentiating characteristic of HBR models pertains to the issue of representation—that is, the manner in which the external world is mapped to the internal processes and states of the simulated human. Most HBR models employ some form of symbolic representation in that human information processes (e.g., input, storage, and retrieval) are simulated by the manipulation of a finite set of discrete symbols. In such

systems, symbols acquire meaning because they refer to specific entities or entity classes in the external world. Symbolic representations are not only congruent with common concepts in linguistics, but also with software programming principles. In contrast, connectionist models provide important nonsymbolic or subsymbolic approaches to representation. Connectionistic models represent internal structures by an interconnected network of links and nodes. In such models, nodes do not represent fixed entities; rather, they acquire their meaning through the continuous adjustment of the links to which they are connected. Compared with symbolic models, these analogue connectionistic systems are less compatible with digital computing systems. Nevertheless, the published literature contains numerous examples demonstrating that digital systems are able to emulate continuous connectionistic systems. Moreover, some HBR models that are primarily symbolic in nature have incorporated nonsymbolic components in the hope of simulating a wider range of cognitive, behavioral, or performance phenomena. Thus, as in the previous discussion of cognitive vs. performance models, the distinction between symbolic and nonsymbolic modeling does not provide mutually exclusive categorization scheme for HBR models.

The present review of HBR models elaborates on two previous ones. Pew and Mavor (1998) provided an extensive review of HBR models. Their work, conducted under the auspices of the National Research Council (NRC) at the request of DMSO, identified 11 architectures that were extant as of 1997. Ritter, Shadbolt, et al. (2002) subsequently conducted a similar study, sponsored by the United Kingdom's Defence Evaluation and Research Agency (DERA), which updated the Pew and Mavor review (through June 2001) by covering 7 additional cognitive architectures not included in the 1998 review.

Table I-1 shows the models reviewed in the two previous studies and the models reviewed present study. The present study reviews 19 different models. It includes most models reviewed in the two previous works but has added 6 new models that have reached sufficient maturity to warrant inclusion.

B. LIMITATIONS

Although I had access to a great deal of published research on the models, I was able to gain only superficial practical experience using the HBR models. Deep

Table I-1. Comparison of the Present Study With Two Previous Reviews of HBR Models

		Model Included in Review?: Yes or No		
Model Name	**Acronym/ Abbreviation**	**Pew and Mavor (1998)**	**Ritter, Shadbolt et al. (2001)**	**Present IDA Study**
Atomic Components of Thought	ACT	Yes	No	Yes
Adaptive Resonance Theory	ART	No	No	Yes
Architecture for Procedure Execution	APEX	No	Yes	Yes
Artificial Neural Networks	ANNs	Yes	No	No[a]
Business Redesign Agent-Based Holistic Modeling System	Brahms	No	No	Yes
Cognition and Affect Project	CogAff	No	Yes	Yes
COGnition As a NEtwork Of Tasks	COGNET	Yes	No	Yes
Cognitive Complexity Theory	CCT	No	No	Yes
Cognitive Objects within a Graphical EnviroNmenT	COGENT	No	Yes	Yes
Concurrent Activation-Based Production System	CAPS	No	No	Yes
Construction-Integration Theory	C-I Theory	No	No	Yes
Distributed Cognition	DCOG	No	No	Yes
Elementary Perceiver And Memorizer	EPAM	No	Yes	No[b]
Executive Process/Interactive Control	EPIC	Yes	No	Yes
Human Operator Simulator	HOS	Yes	No	Yes
Belief-Desire-Intention architecture	BDI	No	Yes	No[c]
Man-machine Integrated Design and Analysis System	MIDAS	Yes	No	Yes
Micro Systems Analysis of Integrated Network of Tasks	Micro SAINT	Yes	No	Yes
MIDAS Redesign		Yes	No	Yes[d]
Operator Model ARchitecture	OMAR	Yes	No	Yes
PSI	PSI	No	Yes	Yes
Situation Awareness Model for Pilot-in-the-Loop Evaluation	SAMPLE	Yes	No	Yes
Sparse Distributed Memory	SDM	No	Yes	No[b]
State, Operator, And Result	Soar	Yes	No	Yes

Notes for Table I-1:

[a] Reasons for excluding ANNs from the present study are discussed in Section I.B.1.

[b] Deemed a “micro” model of cognition, focusing on memory processes.

[c] Reasons for excluding BDI from the present study are discussed in Section I.B.3.

[d] The redesign of MIDAS is discussed in the history and development of the MIDAS model.

understanding of the models is acquired only through extensive use of them. On the other hand, even a superficial analysis of models helps in understanding the breath of the domain.

1. Models Eliminated From Consideration

Even a cursory review of this area reveals a wide variety of models and techniques that could support military simulations. To limit the boundaries of the present review and sharpen its focus, it was restricted to models whose primary purpose is to emulate a wide range of actual human behavior and processes. This restriction eliminated the following classes of models from consideration:

- **Nonhuman models.** Some models incorporate aspects of intelligent behavior and reasoning that are highly constrained in content and maximized in performance. Examples include expert systems for medical diagnoses or computer systems specifically designed to challenge chess grand masters (e.g., Big Blue). Thus, while such models are clearly "intelligent," they are also distinctively nonhuman.
- **Micro-models of cognition.** Computational models of individual cognitive functions, such as memory or auditory perception, are numerous. While these models are useful theoretically, they are too narrowly focused to provide realistic simulations of integrated human behavior and performance.
- **Competence models.** Researchers in psychology and linguistics have developed computational models of linguistic understanding. While these models are interesting, they have limited implications for the overt behavior and human performance representations needed in military simulations.

Also excluded are models that provide a comprehensive simulation of human behavior but pose implementation problems that preclude their incorporation into military simulations. Often, these problems center on computer coding. For instance, the model of cognition and perception developed by Ulric Neisser (1967; 1976) is comprehensive and authoritative. However, this model is largely qualitative and has not been implemented in computer code. Such models were excluded not because they were invalid, but because they were not amenable to computer-based simulations.

Serious implementation problems remain in the models that are included in this review. With respect to the practicality of implementation, the present report probably erred in the direction of inclusion rather than exclusion. In other words, some of the

models undoubtedly have serious computer problems that require special expertise in computer science and artificial intelligence (AI).

2. Alternatives to Rule-based Cognitive Modeling

Most of the reviewed models are rule-based systems for simulating serial and parallel cognitive structures and processes. The elements of these systems are "productions" that take an if-then or condition-action form. Three alternative approaches to rule-based cognitive modeling have received some interest as potential HBRs but are not included in this review: Artificial Neural Net (ANN), Evolutionary Computation (EC), and Belief-Desire-Intention (BDI) models. The exclusion of these approaches from the review is explained in some detail below because of the general interest in these types of models.

a. Artificial Neural Net (ANN) Models

Based on concepts developed in the 1940s (e.g., McCulloch and Pitts, 1943), ANNs are densely interconnected, parallel computing structures intended to emulate the mammalian brain in structure and function. The processing elements of ANNs correspond to the individual neurons of the brain, with weighted associations among elements simulating the synaptic connections. Learning and memory effects are modeled as adjustments to and persistence of those neural weights.

The original purpose of ANNs was to model nervous system functions. As ANNs have evolved, however, the neural network metaphor has become less literal and more figurative. While ANNs may not provide accurate models of biological functions, they have some unique advantages over traditional computing architectures. For instance, it can be shown that ANNs are less susceptible to noise and memory loss than are traditional computers. ANNs have also been successfully used as a statistical technique for predicting complex events.

As models of cognition, ANNs have successfully modeled feature detection processes in perceptual pattern recognition. Not only do these models provide a plausible explanation of perception, but they have also led to the production of practical machine recognition systems. However, human models of perception based on ANNs apply only to perception and thus constitute examples of micro-models of cognition. Also, aspects of

neural nets have been incorporated into more comprehensive rule-based HBR models (e.g., ACT-R,[1] SAMPLE, C-I, and COGENT) to form hybrid systems.

With one notable exception, no one has developed a neural net model as a comprehensive cognitive model. The exception is ART (Adaptive Resonance Theory), which uses neural networks as a centerpiece to explain and model a range of cognitive processes. For this reason, the ART model was included as the only example of an ANN used to simulate human behavior. See Section II.B for more information about ART.

b. Evolutionary Computation (EC) Models

EC refers to a class of computational models designed to solve a broad set of problems. In this sense, EC is similar to an ANN. The unique aspect of EC is its use of the biological concepts of evolution and natural selection as computational metaphors.

Although EC models are able to solve problems, they do so in a nonhuman way. The central notion is to employ a computational model of biological evolution to reach novel solutions that are adapted to the detailed demands of the situation. According to Porto, Fogel, and Fogel (1998), most evolutionary programs can be distilled into the following four-step process:

1. Identifying the population of candidate solutions initializes the program.
2. Existing solutions are then randomly varied through a process of mutation, recombination, or both.
3. Competing solutions are evaluated with respect to a measure of merit.
4. Solutions are selected to determine which will survive for the next iteration. Once the solutions are selected, the process restarts at step 2.

Unlike traditional cognitive modeling, EC does not start with a set of predefined actions or rules or with an explicit knowledge base. Although expert knowledge can be folded into an evolutionary model to make it more efficient, the strength of this approach is its relatively low level of dependence on domain expertise.

---

[1] The Atomic Components of Thought (ACT) is a comprehensive model of cognition developed by John R. Anderson and colleagues at Carnegie Mellon University (CMU). The ACT model (ACT-A) appeared in the fall of 1974, and it has been continuously updated since that time. The current model, called ACT-R, appeared in Anderson (1993), and updated versions of the model have been forthcoming since that time. See Section II.A for more information about ACT.

EC is best suited to difficult problems that possess nonlinear or arbitrary constraints. For instance, it has been applied to difficult problems, such as classifying types of breast cancers, predicting chaotic time series, and deriving novel tactics for tank platoons (Porto, Fogel, and Fogel, 1998). Even though the latter problem would appear relevant, the EC simulation employed in this study was not constrained to behave like humans and was allowed unlimited time to arrive at an appropriate solution. Consequently, it was not included in this review of HBRs.

c. Belief-Desire-Intention (BDI) Models

BDI is a model for developing intelligent software agents based on a philosophical theory of practical reasoning (Bratman, 1987). As implied in the title, a BDI model comprises three essential components:

1. **Beliefs.** Beliefs are some data structure that represents the state of the world. They are generally equivalent to the more common notion of "knowledge," but are based on perceived or "likely" states of the world that may or may not be true at any given moment. Rao and Georgeff (1995) describe beliefs as something that provides information on the state of the system.
2. **Desires.** Desires are information about priorities or payoffs of system objectives. They correspond generally to "goals" as used in rule-based models; however, unlike goals, they are numerous, subject to change, and may be mutually incompatible. Desires represent the motivation underlying the BDI system (Rao and Georgeff, 1995).
3. **Intentions.** Intentions are committed plans of action. The essence of BDI in that it solves the practical problem of choosing a course of action in a dynamic environment. Constantly reconsidering the entire domain of actions at each instant in time is too expensive computationally; conversely, unconditional commitment to a single course of action could result in failure to achieve objective(s). The solution is to focus on a subset of actions corresponding to current desires (committed plans) but be capable of reconsidering these actions at crucial moments. Intentions are the deliberative component to BDI models (Rao and Georgeff, 1995).

The BDI model represents the requirements of an intelligent computational agent. The model is not intended to represent actual human cognitive structures and processes, although the components bear some resemblance to the elements of rule-based cognitive models, such as Soar and ACT-R. Nevertheless, the fundamental motivation in constructing BDI models is to make computationally efficient and effective intelligent

agents—not to model human cognition. For these reasons, BDI models were not included in the present review.

C. ORGANIZATION OF THIS PAPER

Section II summarizes each of the HBR models selected for review. The review focuses on topics that are useful to those who are interested in applying HBR models to military simulations and is organized as follows:

- Model Purpose and History of Development
- Principal Metaphors and Assumptions
- Cognitive/Behavioral Functions Represented
- Applications
- Technical Considerations
- Evaluation.

Section III provides summary comments about HBR models. These comments are brief, which is consistent with the paper's focus on the usefulness and applicability of particular HBR models rather than the overall state of the art in cognitive modeling.

II. DESCRIPTIONS OF HBR MODELS

Table II-1 lists each of the 19 HBR models reviewed in this paper. This table also indicates the Section II paragraph in which the discussion takes place.

Table II-1. HBR Models Reviewed in the Present Study

Section/ Paragraph	Model Name	Acronym/ Abbreviation
II.A	Atomic Components of Thought	ACT
II.B	Adaptive Resonance Theory	ART
II.C	Architecture for Procedure Execution	APEX
II.D	Business Redesign Agent-Based Holistic Modeling System	Brahms
II.E	Cognition and Affect Project	CogAff
II.F	Cognition as a Network of Tasks	COGNET
II.G	Cognitive Complexity Theory	CCT
II.H	Cognitive Objects within a Graphical EnviroNmentT	COGENT
II.I	Concurrent Activation-Based Production System	CAPS
II.J	Construction-Integration Theory	C-I Theory
II.K	Distributed Cognition	DCOG
II.L	Executive Process/Interactive Control	EPIC
II.M	Human Operator Simulator	HOS
II.N	Man-machine Integrated Design and Analysis System	MIDAS
II.O	Micro Systems Analysis of Integrated Network of Tasks	Micro SAINT
II.P	Operator Model ARchitecture	OMAR
II.Q	PSI	PSI
II.R	Situation Awareness Model for Pilot-in-the-Loop Evaluation	SAMPLE
II.S	State, Operator, And Result	Soar

A. ATOMIC COMPONENTS OF THOUGHT (ACT)

1. Model Purpose and History of Development

The Atomic Components of Thought (ACT)[2] is a comprehensive model of cognition developed by John R. Anderson and his colleagues at Carnegie Mellon University (CMU). According to a recent tutorial on ACT (Lebiere, 2002), ACT serves several different purposes: to provide a unified theory of mind, to account for experimental data, to develop educational systems and environments, to design human-computer interfaces, and to interpret data from brain imaging. It is significant to note that, among this diverse set of purposes, Lebiere included the role of ACT in computer-generated forces—that is, to "…provide cognitive agents to inhabit training environments and games" (slide 3, ¶ 5).

ACT evolved from the Human Associative Memory (HAM) model developed by Anderson and Bower (1973). HAM was a connectionist model of semantic memory that represented Anderson's doctoral research at Stanford University. ACT represents the synthesis of HAM and a production system theory of memory (Newell, 1973).

The first ACT model (ACT-A) appeared in the fall of 1974, and it has been updated since that time. The current model, called ACT-R, appeared in Anderson (1993), and updated versions of the ACT-R model have been forthcoming since that time. The current version of ACT-R is 5.0, which is intended to serve as a beta version for 6.0. Version 5.0 differs from the previous one (Version 4.0) in that it includes perceptual-motor components. The ACT-R group at CMU conducts yearly summer workshops and maintains an active Web site at http://act.psy.cmu.edu. Textbooks by Anderson (1976, 1983, 1990, 1993) provide lengthy theoretical explications of the ACT model and its chronological history. A recent article summarizes the history of ACT and focuses on the innovations provided by the most recent version, ACT-R 5.0 (Anderson et al., 2002).

2. Principal Metaphors and Assumptions

Two central concepts in ACT-R are rational analysis and the distinction between declarative and procedural knowledge. According to the rational analysis concept, each

---

[2] Although the use of the acronym "ACT" has been relatively stable, the definition of the acronym has evolved over the years. The original definition was the "Adaptive Character of Thought," which was the title of an influential textbook by Anderson (1990). However, Anderson and Lebiere (1999) published ACT-R 4.0 in a book whose title suggested a new definition of ACT: "Atomic Components of Thought." Others have suggested, perhaps facetiously, that ACT stands for "Anderson's Cognition Theory."

component of cognition in ACT-R is optimized with respect to environmental demands but constrained by computational limitations.[3] Anderson argues that this aspect of cognition is the result of evolution. This rationality is modeled by a cost-benefit model of decision-making—that is, when the model is forced to choose between strategies, it takes the one that maximizes the probability of success while minimizing the costs in computational terms.

Regarding the distinction between knowledge types, declarative knowledge refers to one's stored information concerning facts about the world. In ACT, this knowledge is modeled as a semantic network, not unlike the memory representation in HAM. In contrast, Anderson contends that our knowledge of actions (i.e., how to do something) is quite different. This procedural knowledge is modeled as a production system. Declarative and procedural knowledge are parts of long-term memory (LTM). These two systems communicate through working memory (WM), which is not a separate memory subsystem but rather the subset of knowledge that is currently active.

ACT-R 5.0 incorporates a more modular conception of cognition (Anderson et al., 2002). Its modules are conceived as encapsulated information-processing sites that are devoted to functions, such as identifying objects in visual field, controlling limbs, retrieving information from LTM, and so forth. The modules communicate through limited capacity buffers that place chunks of declarative data (representing the results of their information processing, into buffers that can trigger processing by central production systems). The result of such cognitive processing can, in turn, trigger more information processing in the modules.

3. Cognitive/Behavioral Functions Represented

Input and output (i.e., sensory and motor) functions were primitively modeled in the original version of ACT. A relatively new, specialized version, ACT-R/PM, was developed to provide those needed capabilities. ACT-R/PM was subsequently incorporated into ACT-R 4.0 and its successor, ACT-R 5.0. According to the Computer-Human Interaction Laboratory (CHIL) Web page (http://chil.rice.edu/byrne/RPM/project.html), ACT-R/PM represents

> "…a synthesis of John Anderson's ACT-R theory, Mike Matessa's Visual Interface for ACT-R, and Dave Kieras and Dave Meyer's EPIC

---

[3] The "R" in ACT-R stands for "rational."

> [Executive Process/Interactive Control] system. ACT-R/PM uses the ACT-R production system/Bayesian network as its core cognitive model, and a parallel set of perceptual-motor modules (much like EPIC). The specific modules are derived from the Visual Interface and similar modules in EPIC" [Computer-Human Interaction Laboratory (CHIL), 2002].

Learning is represented at the symbolic and the subsymbolic levels. At the symbolic level, chunking occurs in declarative memory by application of existing rules. New rules are learned through analogy and example. At the subsymbolic level, learning is represented by changes to activation and strength parameters.

Errors and forgetting are products of processes at the subsymbolic level (below individual productions)—namely, activation and strength parameters of the semantic network. Productions may not fire because activations levels are below some threshold. Similarly, incorrect productions may fire because their threshold levels are relatively high with respect to correct productions. Activation and strength parameters also affect the latency of responding.

ACT was originally developed to address cognitive activity and is quite good at simulating individual intellectual functions, such as attention, decision-making, and problem solving. However, as is true for most models of individual cognitive processing, the ACT-R architecture has no representation for affective variables (emotion and motivation) and does not model collective performance.

4. Applications

ACT-R is an academic model explicitly designed to provide a unified theory of cognition and account for experimental data. Nevertheless, ACT-R has had several practical applications, which includes providing the basis for intelligent tutors for math and computer science aimed at secondary education. ACT-R has also been used to model human-computer interaction (HCI) as an HCI design aid, and it has provided a framework for interpreting data from brain imaging. As of September 2002, over 100 published models had been developed in ACT 4.0 (Anderson et al., 2002).[4]

More recently, Anderson has stated that he wants ACT-R to provide computer-generated forces to inhabit training environments and games. An example is work

---

[4] The current version of ACT-R (5.0) is downwardly compatible will thus run all models developed in Version 4.0.

performed by Salvucci, Boer, and Liu (2001) to predict the effects of cell phone usage on driving behavior and vice versa. ACT-R controls a driving simulator and simulated cell phone that represents the use by humans in the same simulation environment.

5. Technical Considerations

A fundamental consideration in implementing ACT-R models is the update rate of 50 msec. This parameter represents a simulation of actual cognitive processing as opposed to a limitation in software or hardware capabilities.

a. Inputs and Input Aids

In terms of input, the user must enter a complete depiction of the initial knowledge state (including declarative and procedural knowledge) and specify all appropriate parameters, which are extensive if the perceptual/motor modules are used. Currently, full exploitation of the ACT-R model requires the modeler to be conversant in LISP (LISt Processing). However, a graphic "stand-alone environment" has been developed for Windows and Macintosh operating systems and permits the non-LISP programmer to create, run, and debug ACT-R models. According to the developers, the tools available in the stand-alone versions are also helpful to the veteran ACT-R modeler in running, inspecting, and debugging ACT-R models. The model and input aids, along with reference manuals for experienced users and tutorials for beginners are available for download at the ACT-R Web site: http://act-r.psy.cmu.edu.

b. Model Output and Analysis Tools

If appropriate, ACT-R provides the capability to model the environment separately from human information processing. ACT-R provides three types of outputs:

1. A trace of productions that fire during execution
2. All changes to WM that occur during execution
3. Items in declarative knowledge that are used in productions.

c. Language and Interfaces

ACT-R is written in Common LISP,[5] and users of the full versions (4.0 and 5.0) must have installed Macintosh Common LISP (MCL) or Allegro Common LISP (ACL)

---

[5] Common LISP is a high-level, all-purpose, object-oriented, dynamic, functional programming language.

for Windows machines before downloading ACT-R. The Windows version of 5.0 is also includes a stand-alone version, which allows the user to run ACT-R without ACL; however, the stand-alone version does not offer all the features of full version [e.g., no compiler, no debugger, and none of the ACL Integrated Development Environment (IDE) tools]. A facility is also available to run limited ACT-R models on the Web.

6. Evaluation

Of all the cognitive models, ACT-R is perhaps the best grounded in experimental psychology research literature. However, despite the developmental tools, users still find ACT-R models difficult to develop, and its parameters are especially difficult to estimate without extensive experimentation.

B. ADAPTIVE RESONANCE THEORY (ART)

1. Model Purpose and History of Development

Adaptive Resonance Theory (ART) is a neural net model designed to explain basic sensory and cognitive processes, including perception, recognition, attention, reinforcement, recall, and WM. The purpose of ART is to model behavioral processes and fundamental brain dynamics. Stephen Grossberg introduced the model in 1976 in a pair of papers in *Biological Cybernetics* (Grossberg, 1976a; 1976b). Grossberg and his colleagues (including, most notably, his wife Gail A. Carpenter) continue to develop the ART model at Boston University's Department of Cognitive and Neural Sciences and the Center for Adaptive Systems.

Several Web sites have been established to promote and support ART modeling, including

- The home page for Boston University's Department of Cognitive and Neural Sciences and the Center for Adaptive Systems, which provides on-line publications relating to ART models written by the principal developers. The Web site address is http://cns-web.bu.edu.
- The "ART Gallery: A Neural Network Simulation Package" developed by Lars Liden, which contains downloads of some ART source code and related documentation. The Web site address is http://cns-web.bu.edu/pub/laliden/WWW/nnet.html.
- The "Adaptive Resonance Theory (ART) Clearinghouse" created by Daniel Tauritz (a member of the faculty in computer science at the University of

Missouri-Rolla), which contains an extensive bibliography and useful introductions to ART modeling. The Web site address is http://web.umr.edu/~tauritzd/art.

2. Principal Metaphors and Assumptions

One of the prime motives for developing ART was to address the stability-plasticity dilemma endemic to ANNs. This dilemma pertains to the fact that while humans are able to retain knowledge of past experience and knowledge (stability), they also are able to respond and change in response to new information (plasticity). ANNs typically represent memory by the relative persistence of weights associated with connections among nodes in the network. The dilemma is that traditional ANNs cannot adapt to new information if the weights hold values indefinitely. On the other hand, ANNs can potentially lose accumulated knowledge if the weights continuously change in response to input.

ART solves the stability-plasticity dilemma by postulating a two-level organization for WM. The bottom, or input, level (F_1) analyzes individual items or features in WM, whereas the top, or output, level (F_2) simulates the functions of superordinate concepts. LTM acts to modulate the weights of directed links between the two levels. Bottom-up signals (F_1 to F_2), modulated through LTM, influence the selection of categories, while top-down signals (F_2 to F_1) exert the effects of expectations on perception and learning. Top-down expectations are compared against stimulus input, resulting in the amplification of input that matches expectations and the suppression of input that does not match expectations. This architecture has two important implications for modeling cognition:

- First, top-down processes modulate attention. Mismatches between top-down expectations and input trigger attention, whereas matches prevent irrelevant stimuli from eroding previously learned knowledge.
- Second, matches between expectations and input result in a reciprocal feedback between bottom-up and top-down systems. The resulting resonance, which is identified with consciousness (e.g., Grossberg, 1995; 1999), leads to long-term modifications in the network (i.e., learning).

ART models are structurally divided into two feedback subsystems. The attentional subsystem is depicted as gain controls that match bottom-up activation with top-down expectations. This subsystem adjusts neural activation to form appropriate categories and expectations and to diminish the effects of noise. The orienting subsystem

is depicted as a single parameter that depicts the sensitivity to mismatches. When this subsystem detects a novel situation, it resets the system to search for a more appropriate recognition code. The latter subsystem is stabilizing because it prevents the network from adjusting to irrelevant stimuli.

3. Cognitive/Behavioral Functions Represented

ART's two-level structure makes it particularly well suited for simulating bottom-up- and top-down-controlled perceptual processes. Several ART models have been developed for modeling perceptual phenomena and illusions, including visual perception, visual object recognition, auditory source identification, and variable-rate speech recognition (Grossberg, 1999).

Learning is driven by the resonance state—that is, adjustments to the link weights occur because of prolonged resonance (activation), not because of momentary, bottom-up sensory activation. Thus, ART resonance serves as an indicator of when the system is receiving data worthy of learning (Grossberg, 2000).

LTM is represented by the weights corresponding to links among nodes in the network. These weights multiply signals in the pathway to determine effects on target nodes. Short-term memory (STM) is depicted as the pattern of activation initiated by bottom-up signals from sensory systems and by top-down signals modulated through LTM links.

Attentional phenomena are modeled by feedback subsystems. The attentional subsystem nonspecifically inhibits all feature detectors that do not receive a suprathreshold excitatory signal based on learned expectations. This effectively makes the system shut off the perception of features that it does not expect and "pay attention" to those features that it does expect. The orienting feedback system, in contrast, detects novel stimuli that signal when the system should break the focus of attention and search for a different and more appropriate pattern of features.

Although ART provides models for sensory and cognitive processes, it is not appropriate for spatial or motor processes. Grossberg (1995, 1999) pointed out that this does not indicate a deficiency of the model; rather, it represents a fundamental difference between sensory-cognitive processes and spatial-motor processes: Catastrophic forgetting would have a devastating effect on sensory and cognitive processes, whereas such forgetting is a good property for spatial and motor representation. Grossberg (2000) pointed

out that if we did not forget, the parameters that controlled our childhood limbs would continue to control our larger and stronger adult limbs. Grossberg explained the differences between sensory-cognitive and spatial-motor phenomena by contrasting mechanisms. In ART, sensory-cognitive learning and memory are based on an excitatory match process that promotes stable learning and inhibits catastrophic forgetting. On the other hand, learning in the spatial and motor realm is governed by an inhibitory mismatch process, which promotes the constant adjustment of gains to sensor-motor maps. Further, the fact that spatial and motor learning does not involve excitatory resonance explains why procedural memory is unconscious (Grossberg, 1999).

Some phenomena of human decision-making under risk are described by affective balance theory (Grossberg and Gutowski, 1987). This particular theory was derived to predict conditioning findings, such as the partial reinforcement effect, based on an opponent process mechanism called the gated dipole. Although this neural network model is related to ART, affective balance theory is treated as a separate model.

4. Applications

Krafft (2002) rightly characterized ART not as a single model, but rather as a family of models, each addressing a different problem in cognition. For instance, the oldest and most basic model is ART1 (Carpenter and Grossberg, 1987a), which was designed to learn and recognize binary patterns. In contrast, ART2 (Carpenter and Grossberg, 1987b) was designed to categorize arbitrary sequences of analog input patterns. ART3 (Carpenter and Grossberg, 1987c) explores parallel search while specifically simulating biological processes (i.e., neural transmission). Finally, ARTMAP (Carpenter, Grossberg, and Reynolds 1991) recognizes arbitrarily ordered vectors of items.

5. Technical Considerations

a. Input and Input Aids

Modelers must completely specify those aspects of the external world that they wish the ART model to recognize and/or learn. For ART, and for other ANNs, the external world is usually a relatively simple stimulus pattern. In contrast, HBR requires little in the way of initial input. Instead, the system "learns" much of the input information through training routines.

Liden (1995) has provided documentation to support the development of ART models. The document is available for download at the Boston University's Department of Cognitive and Neural Sciences and the Center for Adaptive Systems. The Web address for this document is http://cns-web.bu.edu/pub/laliden/WWW/nnet_doc.html.

b. Model Output and Analysis Tools

The output from ART, as from any ANN, is a pattern of activation. This pattern can be mapped to particular responses, thereby simulating choice behavior among a potentially large number of alternatives. Although ART can simulate discrete responding, it cannot currently simulate continuous control. To our knowledge, no specialized analysis tools are available for analyzing ART output.

c. Computer Language and Interfaces

ART can be downloaded from the *ART Gallery* (Linden, n.d.) as "a series of C procedures intended to be called from within other code…" (¶ 1). Three versions of the procedures—UNIX, DOS, and Visual Basic (for use in MS Windows applications)—are available. Although these procedures should, in principle, be relatively easy to interface with other cognitive models and simulations, no examples of such interactions were found.

6. Evaluation

Of all the HBR models reviewed, ART is the only pure connectionist model—that is, it has no symbolic components. Although theoretically pure, this aspect of the model limits its applications to the simplest tasks, such as choice-type behaviors. Anderson et al. (2002) recently argued that while it may be theoretically possible to simulate symbolic processing with a connectionist model, it may not be practically possible to do so. They point out that models that are able to simulate complex behaviors (e.g., continuous control of aircraft) have major symbolic components.

C. ARCHITECTURE FOR PROCEDURE EXECUTION (APEX)

1. Model Purpose and History of Development

The purpose of the Architecture for Procedure Execution (APEX) is to model human performance in complex, dynamic environments. APEX supports a variety of modeling goals, including the development of models to stand in for humans in training

simulations, the evaluation and design of human-machine systems, the prediction of the effects of new technologies on human operators, and the exploration of scientific theories of human performance (Freed et al., 2002). The aspect of APEX that distinguishes it from other models with similar goals is its explicit emphasis on reducing the expertise and time required to develop HBR models.

Michael Freed developed APEX as part of his doctoral dissertation at Northwestern University, which was completed in 1998. APEX continues to be developed by researchers at the National Aeronautics and Space Administration (NASA) Ames Research Center (ARC) and at CMU. Freed and his colleagues are currently trying to build a network of users and maintain Web pages at two sites:

- NASA ARC: http://human-factors.arc.nasa.gov/apex
- CMU: http://www.andrew.cmu.edu/~bj07/apex.

2. Principal Metaphors and Assumptions

At the highest level of abstraction, APEX comprises two major components: an action selection system and a resource allocation architecture. The first major component of APEX, action selection, is based on a Reactive Action Package (RAP) planning scheme, which was developed by Firby (1989) for AI applications. Although RAP was not explicitly intended to model human behavior, knowledge in RAP is represented as tasks (or procedures) organized into a goal hierarchy, a scheme that is consistent with many theories of human decision-making and problem solving. Freed incorporated RAP into APEX because of its ability to interleave the execution of multiple tasks in a complex and changing environment.

The second major component of APEX, the resource architecture, represents the individual elements in the information-processing system, such as perception, cognition, and motor elements. The architecture models the limitations of those information-processing elements, which constrain the action selector system. Current resources that are simulated include those related to cognition, perception, and motor functions. Resource components are articulated to the extent needed for specific applications. For instance, a model of kinesthetic sensing might be required for psychomotor-dominated tasks (e.g., aircraft pilot) but not for cognitively dominated tasks (e.g., air traffic controller). The user has the capability to define additional resources as needed.

Knowledge elements in APEX are represented as procedures defined by the Procedure Definition Language (PDL). PDL is a formalism designed to translate the results from task analyses into a form that APEX can execute. PDL procedures are production-like constructions, which are roughly equivalent to "methods" in the GOMS model.[6] APEX models are based on task analyses where procedures are broken down into smaller components (e.g., "steps") that can embed more detailed steps or call other procedures. The lowest order behaviors in this scheme are termed "primitive actions" and are defined by the PDL language rather than by task analysis. Currently, seven such primitives are defined for PDL (Freed et al., 2002).

Procedures in APEX are not necessarily fixed linear tasks. In APEX, the steps of a procedure can be ordered in three different ways:

1. **Serial order.** This is implemented in PDL by using a "waitfor" clause so that one step is the precondition for the next step in sequence.
2. **Parallel order.** If not under the control of a "waitfor" clause, steps can be executed simultaneously, subject to resource constraints.
3. **Priority order.** The priority of a step for a particular resource can be continuously computed so that order is dependent on the current situation. Shifting priorities can lead to an interruption of ongoing procedures.

3. Cognitive/Behavioral Functions Represented

Currently, vision is the only mode of perception simulated in APEX. Vision is represented as a feature-extraction process, where each visual object is described by a fixed set of feature types: color, intensity, shape, orientation, size, position, motion-vector, and blink rate (Freed, 1998). During visual perception, the visual resource is temporarily blocked. The visual resource model also controls vision by specifying a restricted field of view, variable acuity, and a time lag between sensing and interpreting visual information (Freed et al., 2002). The visual perception process is under the control of the interaction between objects in the external environment and vision resources (i.e., bottom-up processes) and of the long-term knowledge and short-term expectations about the situation (top-down processes). Although the current perceptual model is limited to

---

6 GOMS (Goals, Operators, Methods, and Selection rules) is the best-known engineering model of human information processing and performance (Card, Moran, and Newell, 1983). GOMS is described and discussed in more detail in the context of CCT (see Section II.G).

vision, the potential exists for users to extend the APEX model to other modes of stimulus input.

Output (i.e., motor) functions are initiated by the *start-activity* primitive. For each type of action, a "class" must be defined. This type of action is further defined by two types of "methods," *update-activity* and *complete-activity*, which describe what happens, respectively, during and at the completion of the action. Thus, the user can simulate actions to the level of detail required by the application. Two particular types of outputs (voice and hand) are predefined in terms of existing resources and innate or standard procedures (e.g., *say* and *grasp*) (Freed, 1998).

APEX simulates the performer at a particular level of competence and does not simulate the processes or effects of learning. The model does, however, simulate the simple storage and retrieval of information from memory. The model specifies two types of memory: a visual memory and a long-term semantic memory. Visual memory is a buffer for storing the results of visual processing. The primary purpose of this component is to allow an agent to retain a representation of the visual field after attention shifts to new locations. Semantic memory is a resource that allows storage and retrieval of long-term information. Information in memory is either confirmed, revised, or replaced. APEX does not model memory loss from either decay or through interference.

Decisions in APEX are modeled according to classic decision theory. The process consists of n steps with the first $n - 1$ steps describing information acquisition tasks to obtain relevant information. The nth step is to employ a decision rule that uses the information to choose among alternative responses. The number and type of factors considered may or may not depend on current workload.

APEX is designed for complex, yet highly routinized, procedures—that is, tasks that have a finite number of known actions and outcomes. As a result, problem solving and other open-ended tasks cannot be simulated in APEX.

4. Applications

Although the purpose of APEX is to create a general simulation model applicable to a range of tasks and situations, published research has been limited to a single application: air traffic control (ATC).

5. Technical Considerations

a. Input and Input Aids

The basic input is an analysis of the task(s) to be simulated. GOMS and equivalent task analysis methods appear to be easily translated into APEX code (John, et al., 2002; Freed and Remington, 2000). The principal aid in translating task analyses into PDL code is the *Apex Reference Manual* (Freed et al., 2002).

b. Model Output and Analysis Tools

Sherpa, the graphic user interface (GUI) for APEX, provides a range of services including starting, stopping, and replaying APEX models. With regard to output, Sherpa provides a textual trace of APEX models, showing when individual operators stop and start relative to the initiation time for the entire procedure. These data can then be used to produce PERT charts. Sherpa can also be used to filter data from multiple runs and then direct the results to be saved in a file for analysis off line.

c. Computer Language and Interfaces

APEX is free and available for download from the two Web sites identified earlier.[7] The system runs under a LISP interface. Separate software distributions apply for Mac (OS 9), Windows (2000, 1998), and Linux systems for the current version of APEX (2.2). This version requires MCL or CLISP,[8] according to documentation. The beta version of APEX (2.3) is built on Allegro Common LISP and runs uniformly on Linux, Solaris, Mac (OS X), and Windows (2000) machines.

In current applications of APEX (for ATC), the simulated world is contained within the system. However, the architecture includes essential infrastructure for simulation, trace event logging, and mechanisms for interoperating with external simulations (Freed et al., 2002).

---

7 *NASA ARC* (http://human-factors.arc.nasa.gov/apex) and
CMU (http://www.andrew.cmu.edu/~bj07/apex).

8 Common Lisp is a high-level, all-purpose, object-oriented, dynamic, functional programming language. CLISP is a Common Lisp implementation by Bruno Haible, then of Karlsruhe University, and Michael Stoll, then of Munich University, both in Germany. It mostly supports the Lisp described in the American National Standards Institute (ANSI) Common Lisp standard.

6. Evaluation

The overall goal of APEX is to develop a model that minimizes the time and expertise required to develop models of humans interacting with machines and their environment. It appears that the developers have succeeded, but only for individual tasks of a highly routine nature. Also, of all the models reviewed in the present study, APEX is one of the newest and has not been subjected to the scrutiny of multiple applications.

D. BUSINESS REDESIGN AGENT-BASED HOLISTIC MODELING SYSTEM (Brahms)

1. Model Purpose and History of Development

The Business Redesign Agent-Based Holistic Modeling System (Brahms)[9] is an agent-based simulation tool for modeling the activities of groups in different locations. The original developers of Brahms were Bill Clancey, Dave Torok, Maarten Sierhuis, and Ron van Hoof, who were colleagues at NYNEX Science and Technology located in White Plains, New York. Brahms was first developed from 1992 to 1997, with funding from NYNEX and in collaboration with the Institute for Research on Learning (IRL) in Menlo Park, California. The genesis of Brahms has its roots in a practical problem: implementing a high-speed data line for NYNEX customers. This process required interaction between a sales representative and a trunk supervisor and between those people and critical office equipment (e.g., telephones, faxes). Their interactions were not well represented in off-the-shelf (OTS) simulations, including cognitive-based models of perception and problem solving and business models of work functions. Brahms was specifically designed to represent aspects of teamwork that are not found in most models (e.g., collaboration, "off-task" behaviors, multitasking, interrupt and resume, informal interaction, and geography).

Project funding for Brahms was discontinued when NYNEX merged with Bell Atlantic in 1997, at which point the NASA ARC started to fund Brahms development and applications. The software is owned by IRL, a Silicon Valley-based national research and development (R&D) center that focuses on learning and innovation for schools and the workplace. The software is licensed to NASA and is available for research purposes only. Release of the software for commercial purposes has not been approved. The principals

---

[9] Over the years, the acronym definition has become less relevant. The name is now spelled in upper and lower case, like Soar, and is now used simply as the product name for a modeling language.

(Clancey, Sierhuis, and van Hoof) work for Agent iSolutions, a NASA ARC project team that is developing work systems solutions using agent-based technology. Their Web site address is http://www.agentisolutions.com/home.htm.

2. Principal Metaphors and Assumptions

In some ways, Brahms can be considered innovative because it uses a multiagent simulation to represent interactions among people. In other ways, however, Brahms is quite traditional: Behaviors are organized into hierarchies, and decision-making is represented by production rules. In terms of scope, it is midway between a macro model of organizational functions (such as those represented in business models) and a micro model of cognition (such as Soar or ACT-R) (Sierhuis and Clancey, 1997).

The metaphors and assumptions of Brahms are drawn from three domains: socio-technical systems theory described by Emery and Trist (1960), activity theory introduced by Vygotsky in the 1930s and formalized by Leont'ev (1979) and situated cognition theory by Suchman (1987) and Clancey (1997).

a. Socio-Technical Systems Theory

Socio-technical systems theory is concerned with the relationship between the social system of humans and the technical systems of the modern workplace. Central to the notion of a social system is the concept of human work practices, which are defined as the formal and informal activities of workers as identified by ethnographic techniques. A work practice refers to "…the collective activities of a group of people who collaborate and communicate, while performing these activities synchronously or asynchronously" (Sierhuis et al., 2000, p. 100). This concept of work differs from Frederick W. Taylor's classic conceptions in that work is not defined solely by inputs and outputs, but instead focuses on the workers' actual behaviors and processes, especially those processes related to social interactions among workers. Brahms represents work practices as the actions of autonomous software agents that are expressly designed to capture those interactions.

b. Activity Theory

In a recent publication, Clancey (2002) asserts that most cognitive models incorporate the meta-assumption that cognition and behavior can be described as tasks to be accomplished or problems to be solved. He argues that this assumption is, at best, seriously deficient and proposes that the alternative to task-based analysis of overt and covert

behavior is one based on activities. Clancey defines activities as socially meaningful and located behaviors that take time, and usually involve interaction with tools and the environment. Activity-based analyses allow researchers to account for a variety of important behavioral and cognitive phenomena that are not well captured in traditional models, such as off-task activities (e.g., waiting), nonintellectual motives (e.g., hunger), sustainment—as opposed to active pursuit—of goal states (e.g., playful interaction), and coupled perceptual-motor dynamic actions (e.g., following someone).

According to Clancey (2002), consciously monitored task performance and problem solving is but one of four types of operations, or physical behaviors, that are subsumed by activity theory. The other three are unconscious sequences of performance (i.e., automated processes), ritualized sequences of performance (i.e., scripts), and consciously sustained dynamic relations (e.g., conversations, reading for pleasure, Web browsing). The fundamental unit of activity is the "frame," which is a sociocultural construct that makes behavior meaningful. The two types of frames in Brahms are (Clancey et al., 1998):

1. **Workframes.** The most central representational unit in Brahms is workframes, which are implemented as situation-action rules that describe top-level activities that may potentially be performed by individuals or members of defined groups.
2. **Thoughtframes.** The covert equivalent of workframes is thoughtframes, which model reasoning or thinking processes that have implications for agent beliefs.

c. Situated Cognition

The central concept of situated cognition is that cognitive structures and processes are not independent of the contexts in which they are used. The strong interpretation of this statement is that the influence is so great that symbolic cognitive representations or descriptions are irrelevant and should be replaced by systems that react directly to the environment without the intervention of such descriptions. In contrast to this strong interpretation, Clancey (1993) espouses a weak interpretation of situated cognition that recognizes the value of symbolic models for understanding behavior. He maintains that the mistake made by symbolic modelers is that they confuse the abstract representations of cognition for cognition itself. For instance, although deliberation can be modeled as an internal process that links perception and action, the neurophysiological reality is that it is more like a cycle of coordinated sensory-motor acts. The acts may call on some sort of

stored knowledge, but the very act of remembering "reinvents" the knowledge in light of the new situation. Strong and weak interpretations of situated cognition agree that research (as well as computational models) should be based on ethnographic methods that focus on the analysis of behavior in a real-world context.

3. Cognitive/Behavioral Functions Represented

Consistent with the situated cognition point of view, internal representations of cognition or perception are minimized in Brahms. At the same time, Brahms is a reasonably comprehensive modeling language in which users can embed internal structures and processes as needed. However, the developers caution that incorporating internal components to Brahms agents causes decreased model parsimony and increased computer processing time. Another impediment is that the developers provide no library of cognitive subroutines or advice in developing such components.

The processes and products of perception and attention are modeled through *detectables*. In Brahms, a detectable is a production rule that defines the states of the world (*facts*, in Brahms) that can be noticed by human agents. When detectables are activated, two things happen:

1. The observed facts about the world become incorporated as beliefs of the agent.
2. These beliefs are matched against the conditions portion of the detectable production rule, which may affect the ongoing activity (*workframe*, in Brahms) by initiating one of the following actions: continue, abort, complete, impasse, and end-activity.

Perception is modeled as an errorless process in that facts are translated directly into beliefs.

Actions are also modeled errorless processes. Overt actions occur in the context of a workframe, a rule-based construct that organizes agent activities. A workframe comprises three components: preconditions, which ensure that an activity occurs in appropriate circumstances; actions themselves, which take time and may consume resources; and the consequences of those actions, which are expressed as changes to facts or beliefs.

Only one workframe is active at any one time. However, activity in a workframe can be interrupted at any time to start or continue another one. Then, under the appropriate conditions, the interrupted workframe can be continued at some later time. The intent is to model the situated nature of multitasking behavior: "People are always working on

many different activities, but our context forces us to be *active* in only one. However, at any moment, we can change focus and start working on another competing activity, while queuing others" (Acquisti et al., (2001, p. 4-130).

Stored knowledge (i.e., memory) is modeled as a set of *beliefs*. These are encoded as first-order propositions stating facts that the agent regards as being true or false. Brahms does not provide for second-order beliefs—that is, beliefs concerning other agent's beliefs. In other words, beliefs are local to agents, in contrast to facts, which are general states of the world. Agents start with an initial set of beliefs that are changed as the simulation is executed. Beliefs can be changed as a consequence of actions taken, changes in world states (*facts*) that are detectable by agent, and communications with other agents. Brahms does not distinguish between short- and long-term knowledge storage or states.

Reactive decision-making is modeled in Brahms using *thoughtframes*, which are similar to workframes but entail no overt action. Using production rules, thoughtframes provide the mechanisms through which agents make inferences about the world. In contrast to workframes, thoughtframes consume no time or resources.

Social interactions are explicitly modeled in Brahms in that human agents interact with objects in the world and with other agents. The most direct method of interaction among agents is through *communication*, which is modeled as either requesting or receiving beliefs from another agent. (An agent does not have to request communication to receive it.) In agent-to-agent communication, the message sender and the target are specified. A variant form of communication is *broadcasting*, where an agent transmits beliefs to *all* agents in a predefined area. Like perception, communication is a direct transmission of beliefs that are instantaneous and errorless.

4. Applications

The original applications of Brahms were for modeling business work practices at NYNEX. Two detailed models were created: "front-end" order processing at the Business Network Architecture Center of NYNEX and "back-end" coordination required to make and prioritize assignments of service technicians to jobs (Clancey et al., 1998). To extend the model to a completely different setting, Brahms was also used to describe the coordination of patient visits to an outpatient clinic in a large healthcare maintenance organization.

Since the move of Brahms and support personnel from NYNEX to NASA ARC, the modeling has concerned space-related missions. Brahms has been used to model the following situations:

- Interactions between crewmembers and the Personal Satellite Assistant (PSA), a softball-sized flying robot designed to operate autonomously onboard manned spacecraft in pressurized micro-gravity environments (Bradshaw et al., 2001).
- Operations during the proposed Victoria mission to the South Pole of the Moon that will employ a semi-autonomous robotic rover (Sierhuis, Clancey, and Sims, 2002).
- Offloading activities related to the Apollo Lunar Surface Experiments Package (ALSEP) on the Apollo 12 mission (Sierhuis et al., 2000).
- Activities aboard the Flashline Mars Arctic Research Station (FMARS), a live simulation of a Martian-manned mission conducted in the Canadian arctic (Clancey, 2002).

5. Technical Considerations

a. Input and Input Aids

In contrast to individual cognitive models, which are based on detailed task analyses, the fundamental inputs for Brahms are gained through ethnographic techniques (i.e., observing while participating in the work setting). Video analysis of everyday work settings provide an essential source of data. The Brahms group (e.g., Sierhuis, Clancey, and van Hoof, 1999), from their earliest work, has emphasized the importance of including in the design team the people being studied to provide the primary context for developing and using models. Clancey (2002) performed a variation on this theme when he (the modeler) joined the work team of people being studied for the FMARS project. In that regard, Sierhuis (1996) provided a discussion of ethnographic techniques appropriate for Brahms development.

Input to Brahms is organized by component (sub)models. According to the *Brahms Tutorial* (Acquisti et al., 2001), the typical simulation can be organized into seven component models:

1. **Agent model.** The groups (of people, usually), individual agents (people), and their interrelationships

2. **Activity model.** The activities that can be performed by agents and objects (inanimate artifacts)
3. **Communication model.** The communication among agents and between agents and objects
4. **Timing model.** The temporal constraints and relationships between activities
5. **Knowledge model.** The initial beliefs and thoughtframes of agents and objects
6. **Object model.** The objects (artifacts) in the world used as resources by agents or used to track information flow
7. **Geography model.** The specification of geographical areas and potential paths in which agents and objects perform their activities.

Brahms' models are developed, implemented, and otherwise supported by integrated software known as Personal Agent. This software kit includes Brahms Builder and Brahms IDE (Integrated Development Environment). Both are used to support the development of Brahms models. The older Builder software will eventually be replaced by the newer IDE software, but Builder currently has functionalities that IDE does not yet have, so the developers are supporting both.

b. Model Output and Analysis Tools

Although Brahms could be used to produce quantitative performance measures, the developers suggest that it be used in a qualitative fashion. For instance, Clancey et al. (1998) suggest that the Brahms users should regard its runs as a "theatrical play" that one views and critiques in a holistic and qualitative fashion. To facilitate this process, Brahms includes the AgentViewer application, which is a stand-alone Visual Basic application. AgentViewer parses the results of Brahms run, which have been saved as a MicroSoft Access relational database, and displays it as a two-dimensional (2-D) graphic time line (Acquisti et al., 2001) showing the activities of all agents and objects.

c. Computer Language and Interfaces

Brahms is an agent-based, as opposed to an object-based, programming language (Acquisti et al., 2001). The current version is written in Java. The original version was written in G2, expert systems software produced by Gensym.

Brahms has no interfaces with military simulations or with other cognitive models. However, Brahms has been integrated with CONFIG, a heterogeneous system for

modeling interactive physical components and controls. The integrated Brahms/CONFIG system has been used to simulate the Mars space habitat, where Brahms simulates the physical layout and the work practices pertaining to this environment and CONFIG provides a simulation of the habitat's life support system (Clancey and Malin, 2002). Brahms developers are currently developing a Java-based interface with Adobe Atmosphere to provide three-dimensional (3-D) depictions of simulations.

6. Evaluation

The key strength of Brahms is its ability to model social as well as man-machine interactions in complex environments. The fact that Brahms has modeled complex sociotechnical environments, such as NASA missions, suggests that it would be highly applicable to military scenarios requiring collective action, such as tactical planning and preparing. Also, the unique theoretical position inherent in Brahms allows its models to capture those kinds of work activities that are not well represented in traditional goal-driven cognitive models, such as waiting or following others.

Brahms does not incorporate a detailed model of internal cognitive structure or process. However, proponents note that the open-systems approach allows the incorporation of more sophisticated models of cognition as needed. It would seem that the Brahms model, which is strong on social interactions but weak on individual cognition, would make a perfect match with Soar or ACT-R, which are strong on individual cognition but weak on social interactions. The lack of collaboration is somewhat surprising.

Finally, in some senses, Brahms is a traditional HBR model in its use of software agents and production rules. However, it is very innovative in its incorporation of new concepts of behavior and cognition, such as activity theory and situated cognition. Some readers find the theoretical aspects of Brahms to be thought-provoking and original. Others find the theory overly difficult because it is not well related to extant HBR theories.

E. COGNITION AND AFFECT PROJECT (CogAff)

1. Model Purpose and History of Development

The Cognition and Affect Project (CogAff) is a multidisciplinary project maintained at the University of Birmingham (United Kingdom) by Aaron Sloman and his colleagues in the School of Computer Science and by Glyn Humphreys in the School of

Psychology. The primary purpose of CogAff is to understand how humans and other animals work. A secondary purpose is to discover how intelligent software agents can be improved.

The CogAff group was started in 1991 by Aaron Sloman, in collaboration with Glyn Humphreys. The associated SimAgent toolkit was introduced in October 1994. CogAff and SimAgent were largely based on work that Sloman had done previously at the University of Sussex, where he was Director of Poplog development, a multilanguage AI environment incorporating Pop-11, Common LISP, Prolog, and Standard ML. Some of the earliest work on CogAff dates to a 1981 paper by Aaron Sloman and Monica Croucher. Some of the initial funding for the toolkit was provided by the UK Joint Research Council Initiative on HCI and Cognitive Science (1992–1995), the University of Birmingham (1994–1995), and DERA (1994–1998). More recent support has come from Leverhulme and Sony.

Sloman and the University of Birmingham continue to support the CogAff project. The Web site address is http://www.cs.bham.ac.uk/~axs/cogaff.html.

2. Principal Metaphors and Assumptions

The developers insist that CogAff is not a cognitive architecture per se but rather a conceptual space for describing existing and even potential cognitive architectures. For instance, H-CogAff is an example of a specific architecture derived from CogAff that describes an architecture for human cognition.

The H-CogAff space is described in terms of two dimensions. The first dimension defines three "layers" of cognition, which correspond roughly to stages in evolutionary development:

1. Level 1 is the oldest layer and comprises reactive mechanisms. It is defined mainly negatively since this level lacks the ability to represent and to compare or evaluate future actions and consequences of those actions. These mechanisms are commonly modeled as condition/action rules but could be modeled by neural nets or chemical mechanisms. It can be shown that such mechanisms can lead to complex behaviors but at the cost of explosive storage requirements and/or excessive training or evolution time. Sloman speculates that these mechanisms are largely determined by genetics, with few changes produced by learning.
2. Level 2 is a newer layer that comprises deliberative mechanisms to generate "what-if" inferences. These mechanisms enable planning, predict future

occurrences, and explain past occurrences. These functions require compositional representational capabilities, an associative store of reusable generalizations, and a WM for storing plans and comparing options.

3. Level 3 is the newest layer and comprises reflective or meta-management processes. These processes concern emotions that result from self-observation and self-monitoring skills. Because these skills involve voluntary control of thought processes, they are thought to be uniquely human.

The second dimension, the so-called "triple towers," concerns the three stages of information processing: perception, central processing, and action. The concept is that information flows through the organism from sensors to motors, with some internal processing intervening between input and output (Sloman, 2003). The flow does not always proceed in that particular order, however. It may flow from internal to sensory processing in the case of conceptually driven perception. It may also skip internal processing altogether in the case of highly overlearned reactions.

The triple layers and triple towers combine to form a 3 × 3 matrix. This structure implies that specific information-processing systems have evolved for each of the layers of development. For example, Sloman (2003) speculated that primates have developed perceptual systems for view-invariant object recognition to abstract reasoning. However, an ordered flow through the matrix—such as information entering a low level, proceeding up a hierarchy of abstraction, and then back down for responding—is not assumed. Nor does the model assume subsumption, where higher levels of information processing have control of lower ones. Rather, the architecture is depicted as a labyrinth with many alternative paths.

A central notion of the theory is that emotion is *not* a general state with associated numerical parameters; rather, it is a semantic process—that is, a product of information processing. The theory gives rise to three types or levels of emotions:

- Primary emotions are the products of reactive mechanisms. Sloman (2001) labels these as "proto-emotions" (e.g., freezing, fighting, fleeing, attending, and mating), which result from direct stimulation and have no components of self-awareness.
- Secondary emotions are the result of deliberative processes and can thus be evoked by events that did not happen. Examples include anticipation, apprehension, and hope. Secondary emotions (e.g., frustration) can also be evoked when deliberative processes are interrupted (perturbed) by primary emotions.

- Tertiary emotions are higher order concepts such as adoration and humiliation, which require reflection on the deliberative processes. Tertiary emotions are also caused by interruptions or diversions (perturbances) of deliberative processes.

3. Cognitive/Behavioral Functions Represented

Perceptual and motor functions are represented as stages in the model. Furthermore, just as different types of emotions correspond to the three layers of cognition, different classes of perceptual and motor functions correspond to those levels (Sloman, 2003). For instance, perceptions serving reactive mechanisms may need to be sensitive to fine external detail, whereas perceptions serving abstract deliberative mechanisms must perceive larger patterns (i.e., "chunks") of information. Likewise, motor responses associated with reactive systems may be simple, unmediated, and direct reactions to input, whereas responses associated with abstract systems may be more complex, receiving input from a variety of layers and requiring translation (unpacking) before execution. The exact nature of the perceptual and motor systems within CogAff is not specified, however.

Memory functions also differ with regard to the three layers of cognitive processing. The reactive layer corresponds to stimulus-response behaviors and requires no internal memory. Alternatively, the environment is used as memory. A short-term reusable WM is a key component of the deliberative layer, where the human performs "what if" analyses before responding. In addition to WM, the deliberative layer also uses a content addressable LTM to extend the analyses beyond the immediately perceived world. LTM becomes even more important for the reflective layer because of its abstract quality.

Based on Sloman's (2003) arguments that humans can perform tasks simultaneously, CogAff employs multiple independent agents to model parallel task performance. Although the model does not include features to simulate social phenomena, the fact that the architecture comprises multiple independent agents implies that this capability could be developed.

The model does not include specific modules for learning, decision-making, problem solving, or other cognitive functions. However, with effort, a user could develop these functions within the intelligent agent architecture of the model.

4. Applications

CogAff's developers regard it primarily as a research project and secondarily as an engineering project. Consequently, few practical applications of the project have been attempted. The agents presently occupy very simple worlds. More complexity is planned for the future. At the same time, the developers feel that CogAff with its SimAgent toolkit has several *potential* practical applications, including "...the design of intelligent software of many kinds (e.g., factory controllers, personal assistants, teaching systems, hazard warning systems, aids to managing disasters, etc.) [and the] design of more 'believable' agents in computer games and entertainments" (Sloman, 2002, ¶ 5).

5. Technical Considerations

a. Input and Input Aids

CogAff models are developed and implemented using the SimAgent toolkit. The toolkit was developed in the Poplog software environment, which is an extension of the Pop-11 language and includes several other languages: Common LISP, Prolog, and Standard ML. The toolkit includes three components:

1. **Poprulebase.** This is written in Pop-11, a forward chaining production system that includes a module for developing neural nets and other nonsymbolic systems
2. **The Sim agent library.** This is a Pop-11 extension that permits the design and implementation of reusable extendable software modules
3. **RCLIB package.** This is a Pop-11 extension that is a graphics library for building graphics demonstrations and complex graphics control panels.

The SimAgent toolkit is available from the University of Birmingham School of Computer Science Web site: http://www.cs.bham.ac.uk/~axs/cog_affect/sim_agent.html. The download is free.

b. Model Output and Analysis Tools

The CogAff model does not appear to have specialized output or analysis tools. However, the people associated with Pop-11 and Poplog projects formed Integral Solutions Limited (ISL), which first produced the Clementine Data Mining System in 1994. In fact, Clementine was written mostly in Pop-11.

In 1998, ISL was purchased by SPSS, Inc., a leading supplier of desktop business analysis and data mining software. This alliance integrated Clementine with a suite of numeric and graphic statistical techniques. It would appear that supplementing CogAff with Clementine and SPSS analysis tools would result not only in a desirable product, but also a practicable one to develop.

c. Computer Language and Interfaces

The SimAgent toolkit was developed in the Poplog environment and is written mostly in Pop-11. Software for Poplog is available at a University of Birmingham (United Kingdom) Web site: http://www.cs.bham.ac.uk/research/poplog/freepoplog.html. The software runs on a variety of UNIX and Linux systems. A personal computer (PC) version of SimAgent is available but does not include the graphics component (RCLIB).

6. Evaluation

The potential scope of CogAff is impressive. One of the more interesting aspects of CogAff is the integration of emotional and cognitive processes. Another strong point of the model is that it is based on a long history of computer science applications in intelligent agents.

On the negative side, the developers provide very little in the way of aids for developing CogAff models. This may be the primary reason that practical applications of CogAff to date are few when compared with Soar or ACT-R.

F. COGNITION AS A NETWORK OF TASKS (COGNET)

1. Model Purpose and History of Development

Cognition as a Network of Tasks (COGNET) is a symbolic computational model of cognition. In contrast to many models reviewed in this study, the primary purpose of COGNET is not to test psychological theories; rather, the goal of COGNET is "…to facilitate the cognitive task analysis and description of specific work domains" (Zachary, Ryder, and Hicinbotham, 1998, p. 16).

COGNET was developed by Wayne W. Zachary, an anthropologist and computer scientist, to serve as the focal product for CHI Systems, Inc., a company that he founded in 1985. The model dates to a 1989 report by Zubritsky and Zachary on the description of the requirements of anti-submarine warfare (ASW). The user model in the ASW

application was subsequently used to develop an intelligent, adaptive computer interface for the tasks (Zachary et al., 1990; Ryder and Zachary, 1991).

In 1996, CHI Systems announced the development of the Generator of Interface Agents (GINA), which was a software workbench for developing intelligent agents based on the COGNET theory (Zachary, Le Mentec, and Schremmer, 1996). GINA continues to provide an R&D tool for COGNET. A commercial version of GINA, called iGEN™, is currently being marketed by CHI Systems. Brief descriptions of iGEN™ and COGNET are provided at the CHI Systems Web site: http://www.chiinc.com/.

The most recent version of the modeling system is COGNET-P (Zachary, Ryder, and Le Mentec, 2002). The developers contended that previous versions of COGNET modeled human competency. COGNET-P was specifically designed to model performance and included mechanisms for incorporating time and accuracy constraints. The model also included a metacognition component that accounts for executive control of cognitive processes and self-awareness of some of the processes.

2. Principal Metaphors and Assumptions

COGNET models complex cognitive behavior by assuming that humans are capable of performing multiple tasks simultaneously. The COGNET model is a symbolic computational model that processes information serially; however, at any one time, the model allows several tasks to be in various states of completion, but only one of these tasks is actually executing. Thus, with rapid attention switching, COGNET simulates parallel processing.

The overall framework separates internal processing mechanisms from explicitly represented knowledge. This internal information-processing system, called Blackboard Architecture for Task-Oriented Networks (BATON), is generic since it applies to all types of tasks. BATON is subdivided into three subsystems:

1. A sensory/perceptual subsystem for converting incoming physical data into symbolic information that can be used by the information-processing system
2. An internal cognitive subsystem that constructs and operates on a mental model of the world
3. An action/motor subsystem for manipulating the external world.

The perceptual and cognitive subsystems are linked by a critical information store called “extended working memory,” which subsumes STM, LTM, and WM stores

postulated in most models of human memory.[10] This memory provides a temporary store for all knowledge required to perform the task. The COGNET architecture includes a formal model for representing all forms of task knowledge, which can be subdivided into four parts:

1. A network of tasks expressed as compiled GOMS[11] goal hierarchies for representing chunks of procedural knowledge
2. Perceptual demons that perceive the external world in parallel and "shriek" for the attention of the cognitive processor [incorporated from Selfridge's Pandemonium model of attention (1959)]
3. A multipaneled blackboard for representing and organizing the declarative information relating to the problem
4. Actions that provide mechanisms for having an effect on the world.

3. Cognitive/Behavioral Functions Represented

Perceptual and motor functions in COGNET are not modeled according to a specific theory. Rather, perceptual and motor functions are modeled according to a simulator-centric approach (Le Mentec et al., 1999). According to this notion, which was developed specifically for integrating COGNET agents into simulations, all to-be-perceived information is represented as objects associated with specific perceptual functions that are called whenever an object is made available by the simulation. The user is free to insert characteristics, such as decay from iconic store, as part of those functions. Similarly, actions are simply modeled with two attributes: time to complete the act and physical response resource (e.g., right/left arm or leg). A COGNET metacognitive module tracks response resources. Again, more complicated response phenomena can be modeled if desired.

Although perceptual and motor functions are represented in COGNET, this model is probably better suited for contemplative, open-ended tasks that are not strongly perceptualmotor in nature. For instance, COGNET is inappropriate for fast-paced tasks demanding psychomotor skill, such as tank gunnery. However, COGNET is particularly

---

[10] Zachary, Ryder, and Le Mentec (2002) do not deny that STM/LTM effects exist; rather, they contend that the distinction between the two types of memory stores is unnecessary to model cognitive processes.

[11] GOMS (Goals, Operators, Methods, and Selection rules) is the best-known engineering model of human information processing and performance (Card, Moran, and Newell, 1983). GOMS is discussed in more detail in the context of CCT (Section II.G).

well suited to modeling complex time-constrained, multitask situations that require performers to switch the focus of their attention repeatedly.

The rule-based task knowledge components of COGNET provide the capability to model cognitive processes, such as problem solving and decision-making. The semantic frameworks instantiated in COGNET's blackboard provide the means with which the system acquires and maintains SA (Hicinbotham, 2001).

Metacognitive functions are included in COGNET-P, the most recent version of COGNET. These functions permit the user to model self-awareness phenomena, such as subjective workload (Zachary, Ryder, and Le Mentec, 2002). The metacognitive control functions, together with the ability to model independent cognitive agents, also provide the capability, at least in principle, to model coordination among multiple team members (Zachary and Le Mentec, 2000).

4. Applications

The original applications of COGNET were in vehicle-tracking tasks in the context of ASW (Zubritsky and Zachary, 1989) and ATC (Seamster et al., 1993). COGNET has since been used commercially to design interfaces for telephone operators (Ryder et al., 1998) and as the basis for an intelligent tutoring system embedded in a shipboard tactical air defense system (Zachary et al., 1999).

5. Technical Considerations

a. Input and Input Aids

COGNET input and output is managed through the iGEN™ set of software tools, which include the following capabilities:

- The COGNET Graphical Representation (CGR) model and other tools for creating and editing user models
- Tools for debugging and testing user models
- A module for translating the CGR model to a COGNET Executable Language (CEL), which can be run on the execution engine and compiled as an intelligent "kernel."

b. Model Output and Analysis Tool

A fourth iGEN™ capability is a communications "shell." This shell includes an application programming interface (API) that links the compiled kernel with the specific application environment, which may be a training simulation or an external environment created in COGNET. This shell also includes the capability to script interactions between the BATON cognitive engine and the environment.

c. Computer Language and Interface

COGNET and iGEN™ were written in C/C++ and can be run on a variety of platforms, including Microsoft Windows, Macintosh, and UNIX-based operating systems. According to Emmerson and Nibbelke (2000), proficiency in iGEN™ modeling is difficult, but the system provides more help and documentation and is more user friendly than comparable cognitive models.

6. Evaluation

Of the cognitive models, COGNET is perhaps one of the more practical ones because it is focused on engineering applications as opposed to theory building. One of the principal advantages of COGNET is its "shell," which allows users to interface with other applications, including models and simulations. As pointed out by Emmerson and Nibbelke (2000), the COGNET model and other applications can reside on different machines that communicate through the Transmission Control Protocol/Internet Protocol (TCP/IP) or an analogous procedure.

G. COGNITIVE COMPLEXITY THEORY (CCT)

1. Model Purpose and History of Development

In a 1985 paper, David E. Kieras and Peter G. Polson (1985) introduced Cognitive Complexity Theory (CCT) as an elaboration of the Card, Moran, and Newell's (1983) concepts of GOMS (Goals, Operators, Methods, and Selection rules) and the MHP (Model Human Processor). The purpose of CCT is to provide the theoretical basis for an engineering mode of interface design.

CCT provides the theoretical base for a GOMS analysis tool, called Natural GOMS Language or NGOMSL. Several similar analysis tools, such as the Keystroke-Level Model, Card-Moran-Newell (CMN) GOMS, and Cognitive-Perceptual-Motor

(CPM) (John and Kieras, 1996), derived from GOMS are used to support the design of interfaces. NGOMSL differs from the other tools in that it is based on an explicit model of cognition, albeit a simplified and highly constrained one, that enables such tools to predict performance, learning, and transfer phenomena.

The simple CCT model provides the theoretical basis for NGOMSL. In contrast, the follow-on executable forms of NGOMSL—GOMS Language (GOMSL) and, the more recent version, GOMS Language Evaluation and Analysis-3 (GLEAN3)—are based on the more complex and comprehensive EPIC cognitive architecture (Kieras, 1999). Nevertheless, CCT remains a simple and useful model of cognition that represents human performance on certain types of serial tasks.

2. Principal Metaphors and Assumptions

CCT is based on Card, Moran, and Newell's (1983) concept of GOMS, which models human performance using four major constructs: Goals, Operators, Methods, and Selection rules. The GOMS and CCT models assume that human action is oriented and organized to achieve specific *goals*. The elemental actions required to reach those goals are *operators*. Operators, in turn, are organized into related sets called *methods.* Operators can refer to elemental perceptual or motor acts or may call other methods that are oriented to subordinate goals of performance. Thus, human performance is organized to satisfy a hierarchical structure of goals and subgoals. The fourth GOMS construct, *selection rules*, is invoked when multiple methods exist for accomplishing a specific goal.

The CCT model comprises two major components (Bovair, Kieras, and Polson, 1990): a *device model*, which is modeled as a generalized transition network (Kieras and Polson, 1985), and a *user model,* which is the more important component that models the user's knowledge as task methods and operators. CCT represents these user model elements as production rules, with the stipulation that they be executed in strict sequential order. In addition to representing external primitive acts, productions also alter the contents of WM. According to the theory, the productions are the fundamental units of cognition. Productions are also assumed to be equally difficult to learn and can be transferred to a new task at no cost in task performance. Task "complexity" is defined by counting individual productions, which can be done at least three ways:

1. Number of production rules in LTM (index of learning difficulty)
2. Number of cycles in WM (index for difficulty of use)
3. Number of shared productions among systems (index of transfer).

3. Cognitive/Behavioral Functions Represented

CCT represents sequential task performance only and, consequently, does not incorporate many complex cognitive or behavioral functions. Perceptual and motor functions are simply represented by *external* operators that exchange information between the simulated user and simulated environment. Perceptualmotor phenomena, other than basic temporal relations, are not represented. Likewise, attention and situational awareness are not represented in CCT. Kieras and Polson (1985) explicitly excluded cognitive processes from the model because they felt that the large increase in model complexity would not be accompanied by a corresponding gain in understanding the cognitive complexity of interface design.

Nevertheless, CCT provides several "built-in" *mental* operators to represent covert cognitive processes. For instance, CCT has mental operators for aiding the basic decision-making process, for setting up a new goal, and for storing and retrieving elements from memory. However, CCT does not provide a representation for acquiring these elements in the first place—that is, it provides no model of learning.

With respect to memory, CCT distinguishes between LTM and WM. LTM is represented as a set of production rules. WM, in contrast, is represented declaratively as a list of attributes and values. WM can be directed to drop information ("forget" command), but information is not automatically lost from WM. Also, WM capacity is not explicitly limited. The developers advise users to examine WM contents and exchanges during model runs to determine cognitive load, which is defined as the number of extant goals in WM at any point in time.

4. Applications

Most applications of CCT and NGOMSL have been related to HCI. Initially, CCT was used to evaluate the cognitive complexity of text editing and menu systems (e.g., Polson, Muncher, and Engelbeck, 1986). NGOMSL was subsequently used to study the interactions of humans and full-sized Macintosh software for the computer-aided design (CAD) of factory workstations (Gong and Kieras, 1994; Kieras, Wood, et al., 1995).

More recently, GLEAN has been used to generate initial estimates of task times for an air defense officer operating a Navy Aegis watch station (Freeman et al., 2002). GLEAN was used early in the design process, employing a notional interface. Time

estimates were then passed onto other more complex models to further refine the design requirements.

5. Technical Considerations

a. Input and Input Aids

CCT models require the user to provide three categories of information in structured formats:

1. A set of task instances that describe the performance requirements, including relationships among task methods and goals
2. A specification of general and specific knowledge required to interact successfully with the interface expressed in a modified production rule format
3. An abstract representation of device states and transitions.

One of the explicit purposes of developing CCT and NGOMSL was to facilitate human model development. Kieras (1994) noted two problems with related models. First, production-system computer languages [e.g., the Parsimonius Production System (PPS)] are difficult to use because of their excessive abstraction and detail (much like an assembler computer language). Second, task analyses are not performed in a standardized and reliable manner. NGOMSL was developed to address both issues, and, although its features were developed especially for CCT, NGOMSL can be used to develop any sort of GOMS model. With regard to user support, Kieras (1994, 1996) has produced two manuals that detail the procedures for developing models of HCI using NGOMSL.

b. Model Output and Analysis Tools

GOMSL, the computer-executable variant of NGOMSL, is designed to be used with a software utility called GLEAN (GOMS Language and Analysis). The original version of GLEAN was explicitly based on CCT (Kieras, Wood, et al., 1995), but the newest version, GLEAN3, is grounded in EPIC, which is a more comprehensive cognitive architecture than CCT (Kieras, 1999). (EPIC is discussed in some detail in Section II.L of this paper.) Downloads of GLEAN and support documents are available at David E. Kieras's GOMS Web site: http://www.eecs.umich.edu/~kieras/goms.html.

GLEAN provides detailed output regarding model execution, including the capability to trace the sequence of methods (steps) executed, the contents of WM, and the

state of the simulated device. In addition, the final output includes the frequency, average time, and total execution time of each operator and method (Kieras, Wood, et al., 1995).

Based on formulae derived by Kieras (1994), GLEAN also provides some performance predictions, including the following:

- Task execution time, based on the total number of productions to be performed
- Procedure learning time, based on the number of new productions to be learned
- Transfer of training, based on the number of productions shared by the original and transfer tasks
- Time to memorize, based on the number of chunks required to learn.

c. Computer Language and Interfaces

The original version of GLEAN was implemented in Macintosh Common LISP. The newest version (GLEAN3) is written in C++. GLEAN3 models can be composed to operate two in two modes. The first mode is like a stand-alone simulation tool, where device functions are simulated within the GLEAN system. This mode is the standard version and requires the user to know NGOMSL. The second mode is an advanced mode, where the user employs a library of class definitions that support developing multiple, fully interactive devices and multiple humans. No additional programming support is available for this mode, which requires that the user be fully conversant in C++ object-oriented programming.

6. Evaluation

Based on the popular MHP and GOMS, CCT provides a simple, concise, and face-valid model of human cognition. However, its simplicity leads to problems. First, the CCT model is limited to a constrained category of tasks: man-machine interface procedures that are strictly sequential. Second, important aspects of cognition, especially input and output processes, are not modeled in any detail. Both of these problems are addressed in GLEAN3 (Kieras, 1999); however, the underlying model for GLEAN3 is the more complex EPIC model, not CCT.

Parallel comments can be directed toward the modeling tools. The GLEAN code is conceptually simple and well documented. On the other hand, the code is a bit cluttered

with "housekeeping statements" (e.g., responses to procedure calls, declarations, returns), which have no psychological significance.

H. COGNITIVE OBJECTS WITHIN A GRAPHICAL ENVIRONMENT (COGENT)

1. Model Purpose and History of Development

Cognitive Objects within a Graphical EnviroNmentT (COGENT) is a modeling system for developing and exploring models of cognitive processes. It was developed by Rick Cooper, John Fox, and their colleagues at the School of Psychology, Birbeck College (London) and at the Advanced Computational Laboratory, Imperial Cancer Research Fund.

COGENT evolved from a project named "Abstract Design Rendering Executable Models" (AD REM) that spanned from October 1990 to October 1995. The purpose of AD REM was to develop a standardized method to model cognitive structures and processes. The principal outcome of that project was an object-oriented version of Sceptic, a high-level simulation language for modeling cognition (Cooper, 1995). The COGENT project started in 1996 upon the completion of AD REM and is presently supported by the United Kingdom's Engineering and Physical Sciences Research Council.

Downloads of COGENT are available at a Web site maintained by the School of Psychology at Birbeck College (London): http://cogent.psyc.bbk.ac.uk. Version 2.1 was released on October 26, 2001, and a beta release of Version 2.2 became available on September 20, 2002, and continues to be developed. Both versions are available in two releases: a student release, which can be downloaded for free and used for academic and noncommercial use, and a professional release for other uses.

2. Principal Metaphors and Assumptions

In contrast to models such as Soar and ACT-R, the sole intent of COGENT is to provide tools for cognitive modeling, not to provide an overall architecture for cognition (Cooper, Yule, and Sutton, 1998). Although relatively atheoretical, COGENT makes a few innocuous assumptions about cognition. First and foremost, COGENT assumes the principle of "functional modularity"—that is, the belief that intellectual functions can be described as the interaction among semi-autonomous subsystems (Cooper, 1995).

The fundamental components of COGENT are low-level processing components or objects. Each of these components is completely configurable in that different capabilities can be assigned to different instances of objects in the same category. For instance, STMs and LTMs are modeled by separate buffers with different characteristics and parameter values. Including memory buffers, COGENT is divided into five basic components (Yule and Cooper, 2000):

1. **Memory buffers.** General-information storage devices that can model a variety of STM and LTM effects.
2. **Rule-based processes.** Objects that contain sets of rules, formatted like production systems, for processing information.
3. **Connectionist networks.** A nonsymbolic or subsymbolic system for modeling spreading activation, feed-forward, learning, and other processes.
4. **Input sources/output sinks.** Although not formal parts of cognitive models, these components allow the modeler to control the model and determine outcomes from it.
5. **Inter-module communication links.** Connections that enable communications among other components through read or write operations.

3. Cognitive/Behavioral Functions Represented

Memory is modeled as buffers, which are general-information storage devices that can be used for short- and long-term storage. Buffer properties specify capacity limitations, decay parameters, access restrictions, and other characteristics—as specified by the user.

Other cognitive functions are not pre-specified in COGENT. The modeler must build processes as appropriate. For instance, COGENT has no pre-specified learning mechanism; however, one could be contrived through the use of appropriate parameters in the connectionist network.

4. Applications

Because of the developers' association with cancer research, several COGENT models have been built around medical diagnosis tasks and skills (e.g., Cooper and Fox, 1997). In the tutorial for Cognitive Science Society, Yule and Cooper (2000) built a demonstration model using the assumptions of the so-called "modal model of memory"

(Atkinson and Shiffrin, 1968), which displayed recency and primacy effects. It has also been used to model concept learning in children and HCI.

5. Technical Considerations

a. Input and Input Aids

The central idea of COGENT is to present a visual programming environment for creating, editing, and testing cognitive models. As described by Yule and Cooper (2000):

> The programming environment allows users to develop cognitive models using a box and arrow notation that builds upon the concepts of functional modularity (from cognitive psychology) and object-oriented design (from computer science) (p. 2).

In addition to these inherent visual aids, COGENT also has a "research programme manager" who keeps track of the evolution of the model as it is developed by iterative improvements. The model history is depicted as a graphic time line with branches.

b. Model Output and Analysis Tools

The user can view, in real-time, changes to the contents of components as the model operates. Model outcomes can be quantitative or qualitative. For instance, COGENT can be programmed to display the current state of a model playing the Tower of Hanoi. In addition, the COGENT package includes advanced data visualization tools (tables, plots, and histograms) to analyze model outputs. Finally, COGENT includes a Model Testing Environment, which can be used to schedule experiments in which "subjects" are run in blocks of trials that systematically vary selected parameters.

c. Computer Language and Interfaces

Version 1 of COGENT was developed in Prolog for UNIX systems. Version 2.X runs on MS Windows 95/98/NT (and possibly Windows 2000) and on various versions of UNIX. An Apple Macintosh version also runs in the X windows server of OS X.

6. Evaluation

On the positive side, COGENT is remarkably user friendly and can conform to a wide variety of cognitive structures and processes. However, because the system is atheoretical, it forces the user to provide or develop a detailed model of cognitive processes

that may not be well understood. Also, no evidence exists of the widespread adoption of COGENT or of examples of COGENT interactions with other models or simulations.

I. CONCURRENT ACTIVATION-BASED PRODUCTION SYSTEM (CAPS)

1. Model Purpose and History of Development

The Concurrent Activation-based Production System (CAPS) was developed by Marcel Just, Patricia Carpenter, and their colleagues at CMU. The original version of CAPS was a simple production system for modeling reading (Thibadeau, Just, and Carpenter, 1982). A unique aspect of the original model was that it incorporated subsymbolic aspects (spreading activation) into the symbolic production system representation. The next generation of CAPS—Concurrent, Capacity-Constrained Activation-Based Production System (CC CAPS, or simply, 3CAPS)—introduced the concept that the total amount of activation is constrained and that the total value varies among individuals (Just and Carpenter, 1992). The most recent computational model, 4CAPS, is organized into collaborative modules, which are intended to correspond to the functions of different cortical areas (Just, Carpenter, and Varma, 1999).

The Center for Cognitive Brain Imaging (CCBI) at CMU is responsible for CAPS development. The Office of Naval Research (ONR) funded workshops to disseminate 3CAPS in 1995 and 1996. The current thrust is in modeling brain function; however, the theory is undergoing further development. The Web site for CAPS and other projects at the center is http://coglab.psy.cmu.edu/.

2. Principal Metaphors and Assumptions

The metaphors and assumptions are best understood by describing them in the context of the model's development. CAPS, even in its earliest form (Thibadeau, Just, and Carpenter, 1982), was conceived as a hybrid model incorporating a symbolic production system and subsymbolic activation values and an associative network. Like Construction-Integration (C-I) theory (see Section II.J), this early version of CAPS focused on reading comprehension and made the following assumptions about human cognition:

- Elements in WM (facts) have value attributes (activations) that reflect their strength or the degree to which they are believed.
- An element can cause a production to fire if it matches symbolically the conditions component of the production and the activation value exceeds a specified threshold.

- Cognitive processing is represented by production firings, which cause activation to be propagated. The flow of propagation proceeds from one WM element (called the source) multiplied by a factor (called the weight) to another element (called the target).
- Processing is explicitly parallel—that is, no limit exists for the number of productions that can fire on the same cycle, and no explicit mechanism exists to resolve conflicts.
- LTM (a declarative knowledge database) exists separately from WM.

The next iteration of the model (3CAPS) added the constraint that the total amount of activation in WM was capped at some specified value (Just and Carpenter, 1992). This total activation can be used to keep elements active in WM or to propagate activation by firing production rules. If the system capacity is exceeded, elements in WM may be forgotten and/or productions may have to be fired several times to activate, thereby slowing down performance. The actual constraining value varies from person to person, accounting for individual differences in WM performance.

The most recent version of the model (4CAPS) is designed specifically to model brain activation patterns involved in high-level cognition (Just, Carpenter, and Varma, 1999). These patterns are represented by modules of production systems that model specific areas of the brain, such as Broca's area, Wernecke's area, and the dorsolateral prefrontal cortex. Just as with these brain areas, the 4CAPS model recruits the appropriate module to perform a task but modifies the recruitment if the task changes in size or in quality of its demands. The primary output of 4CAPS is the location and amount of processing per unit time, which is designed to predict the pattern of brain activity recorded by technologies such as functional magnetic resonance imaging (fMRI) and positron emission tomography (PET).

3. Cognitive/Behavioral Functions Represented

CAPS models represent central cognitive functions (e.g., comprehension) and do not include peripheral functions, such as perceptualmotor acts. All knowledge is encoded as productions in long-term storage, and no mechanisms to acquire or modify that knowledge (i.e., learning) are included in CAPS.

One function of attention is to allocate resources among cognitive processes. The 3CAPS model implements the following allocation strategy: When capacity limits are

exceeded, storage and activation processes are decremented in proportion to current demands. No other allocation scheme or function of attention is modeled.

Memory structures and processes are explicitly defined in CAPS. LTM includes procedural and declarative components. WM, in contrast, is exclusively declarative—that is, it contains only facts. Forgetting is modeled by decrementing the activation values of "old" elements that remain in WM from cycle to cycle without receiving explicit activation. Just and Carpenter (1992) characterized this as "…continuous forgetting by displacement" (p. 135).

CAPS provides the ability to simulate simple problem solving and decision-making behavior as comprehension-like processes.

4. Applications

To date, most applications of CAPS have been limited to experimental or laboratory tasks designed to tap specific cognitive structures and/or processes. For instance, Just et al., (1996) developed a sentence comprehension task that systematically varied difficulty and correlated it with the pattern of brain activity measured by fMRI. Just, Carpenter, and Varma (1999) later modeled the behavioral results and the pattern of brain activations using 4CAPS. Goel, Pullara, and Grafman (2001) have also used 3CAPS to model the performance of normal performers and patients with lesions to the prefrontal cortex as they attempted to solve the Tower of Hanoi puzzle (a popular, experimental problem solving task). Hegarty (2001) used 3CAPS to model a mechanical reasoning task (mental animation), where respondents predict the direction of rotation of a particular component in a complex system of interacting objects (e.g., rope and pulley).

The apparent exception to the use of CAPS for academic applications is Kirlik and Byrne's (1994, reported in Byrne, 1994) development of a "situated" CAPS model (S-CAPS) for representing a "real-world" task: performing a preflight check of aircraft systems. The model requires the simulated performer to make safety checks, which are interrupted by a critical message from the control tower. The pilot is required to comprehend the message and determine whether to reply and then to continue completing the safety checks. The model computed the number of decide-act cycles required to complete the safety checks under three conditions: with no checklist, with a checklist but no pen, and with a checklist and pen. The findings indicate that external support (i.e., checklist and pen) reduced the WM requirement and the number of cycles to complete the safety check.

5. Technical Considerations

As with many other models reviewed in this paper, CAPS has a fixed update rate of 50 msec.

a. Input and Input Aids

The user must specify all elements of the task or problem and their interrelationships. For instance, in the mechanical reasoning example, Hegarty (2001) had to parse the mental animation task into individual strands and components that represent elements in WM. Each element is represented as a proposition that can be recognized in the conditions component of production rules. Then, the initial activation values of all elements (ranging between 0 and 1) and the weights corresponding to relations between pairs of elements must be specified. The user must also specify the threshold for activation and the maximum total activation value allowed.

Documentation and technical support are available through Sashank Varma's Web site at Vanderbilt University:
http://peabody.vanderbilt.edu/ctrs/ltc/varmas/varmas.html.

b. Model Output and Analysis Tools

The basic outputs from CAPS are activation values of each element per cycle. Activations can be translated into choice behavior if the choices are defined as specific patterns of activations. If the "correct response" is so identified, statistics such as the number of processing cycles required to reach that state can be calculated. Another type of outcome is to determine the tradeoff between processing cycles and constraints for a specific model. However, the user must generate the results "by hand" because no analysis tools for CAPS currently exist.

c. Computer Language and Interfaces

3CAPS is implemented in Common LISP for the Apple Macintosh. Two versions currently exist: Version 3.1 for 68040-based Macs and Version 4.1 for PowerPC-based Macs. The CAPS software has not yet been interfaced with other cognitive models or military simulations.

6. Evaluation

An outstanding feature of CAPS is that it represents processing capacity in a theoretically plausible and empirically valid manner. CAPS is also unique in its systematic representation of individual differences. As a practical model of cognition, however, CAPS is hampered by a lack of real-world applications. To implement CAPS in military simulations would require considerable development, and such a developmental effort would be handicapped by a lack of documentation.

J. CONSTRUCTION-INTEGRATION (C-I) THEORY

1. Model Purpose and History of Development

The Construction-Integration (C-I) theory was developed by Walter Kintsch and his colleagues at the Institute of Cognitive Science (ICS) at the University of Colorado. The original model was derived from a theory of sentence processing by Kintsch and Dutch linguist Teun A. van Dijk in the late 1970s and early 1980s (e.g., Kintsch and van Dijk, 1978). Kintsch (1998) has since argued that the comprehension process provides a general paradigm for cognition.

Two important extensions of C-I Theory are currently under development at ICS or allied organizations at the University of Colorado. The first is a model of behavior using novel computer interfaces developed by Muneo Kitajima and Peter G. Polson (1997) called LInked model of Comprehension-based Action planning and Instruction (LICAI). This model was derived from C-I Theory to model the ability to comprehend incomplete instructions from systems with GUIs. These same researchers developed a related model for exploring new Web sites, which they have labeled Comprehension-based Linked model of Deliberate Search (CoLiDeS) (Kitajima, Blackmon, and Polson, 2000).

The second development is that LICAI and the general C-I model have incorporated some of the concepts of Latent Semantic Analysis (LSA). LSA is an automated, corpus-based system for deriving meanings of individual words, sentences, paragraphs, and entire essays. It transforms word usage patterns into an *n*-dimensional semantic space. The meaning of any item is a vector in this space. LSA has been used to explain the disproportionately rapid rate of language acquisition in children given the apparent paucity of stimulus input. The LSA network has been used to model the structural

relationships in LTM as posited by C-I. Kintsch (1998) has shown reasonable results from LSA when a spreading activation model is used to link items.

All three models are under active development at the University of Colorado. Links to the three models can be obtained through the University of Colorado ICS Web site: http://psych-www.colorado.edu/ics.

2. Principal Metaphors and Assumptions

C-I Theory, LICAI, and LSA are closely interrelated models but differ somewhat in their metaphors and assumptions. Therefore, each is discussed separately below.

a. C-I Theory

At the most general level, C-I Theory differs from other HBR models derived from Newell and Simon's General Problem Solver (GPS) in two important respects:

- First, the representational unit is a proposition, not a production. In C-I Theory, a proposition is defined as the smallest unit of meaning to which a truth value can be attached. It usually corresponds to a component of a sentence, such as an individual phrase or clause. Propositions are normally represented as an ordered tuple consisting of a predicate and associated arguments. These arguments include the specific action to be performed and the object(s) of the action. Propositions are linked (associated) by their shared arguments—that is, the strength of the associations is a function of the number of arguments that the nodes have in common.
- The second fundamental difference between C-I and traditional Newell-Simon models is the central cognitive process. For Soar and ACT-R, the central process is problem solving. In contrast, the central process in C-I Theory is comprehension. One implication of focusing on comprehension is that the C-I model stresses goal formation, as opposed to goal use in ACT-R, Soar, and related models. The goal formation process in C-I Theory guides all internal search processes.

As implied by the title, C-I is a two-stage process. During the first stage (construction), approximate, inaccurate representations are formed via weak (i.e., general or context-insensitive) rules. These rules are represented as productions that do not result in precise inferences; rather, the rules are relatively sloppy and result in an initial output that is incoherent and perhaps even contradictory (Kintsch, 1998). The primary purpose of such rules is to generate a network of associated propositions that are associated with the input; however, because the rules are sloppy, some of the associates will be closely

related to the target meaning, and some will be much more remote. The number of retrieval attempts, k, is a parameter of the model and is usually set between 5 and 7. Thus, construction is conceived as a bottom-up process in that it is primarily data-driven.

The tentative representations generated during the construction stage are assimilated in the second (integration) stage. This process constrains comprehension through a spreading activation process that is propagated through the network of propositions, proceeding from the source (goals, context) to the target (potential responses). Nodes that are mutually reinforced associations are strengthened and elaborated, while weak unsupported ones are pruned. Activation continues until the interpretation of the input is clear or until it is clear that continued activation provides no further clarification. Thus, integration is portrayed as a top-down, constraint-satisfaction phase.

One replication of these two stages is a "cycle," which is the fundamental temporal C-I unit of comprehension. The purpose is to converge on the "correct" interpretation of the input—that is, a stable state in which the nodes that are meaningfully related to the target are activated and the nodes that were originally retrieved but not related to the target are suppressed. For complex input (e.g., connected discourse), comprehension is a series of C-I cycles, where a subset of the results from one cycle are carried over to the next cycle. This aspect of the model is dependent on two more parameters, in addition to k (the number of retrieval attempts):

1. n_i, the size of the "chunk" (i.e., in number of propositions) comprehended on cycle i. This parameter varies as a function of the complexity of input and the competence of the performer. In examples provided by Kintsch and van Dijk (1978), this parameter varied between 7 and 12 in a single example of connected discourse.
2. s, the subset of n_i propositions that are stored in an STM buffer to be carried over to the next cycle. This parameter could also vary as a function of input and performer but is usually set at a small constant value, such as 4.

b. LICAI

In much the same manner that a skilled reader constructs a contextually appropriate interpretation of text, a skilled computer user selects actions based on his/her comprehension of the current situation. The situation is defined by the user's current goals and expectations, the information on the display, and the information retrieved from LTM—including facts about objects on the display and task procedures.

In accord with C-I Theory, HCI is conceived as a two-stage comprehension process. During the construction phase, users generate propositions representing the three aspects of the situation. The propositions are interrelated by a network defined by the overlap in proposition arguments. Analogous to the original C-I Theory, the construction stage in LICAI generates alternative action sequences and the knowledge necessary to choose among them. During the integration phase, a spreading activation process reinforces relevant knowledge nodes and suppresses irrelevant ones. The process ensures that the action selected is appropriate to the current situation.

In the context of solving arithmetic word problems, Kintsch (1988) recognized that solutions require the recognition and specification of appropriate procedures that have their own domain-specific structure and syntax. This required the addition of special-purpose strategies that were represented as production rules. Similarly, actions in LICAI are represented as action-object pairs (Kitajima and Polson, 1995). This representation has three components:

1. A proposition that specifies the action performed, display object on which to be acted, and the object's functions
2. A set of conditions expressed as one or more propositions
3. Another set of propositions that are added to the network representing the results of the action.

Propositions related to action-objects are interrelated with other propositions (e.g., task and device goals, display objects, information retrieved from LTM) in a network that acts to constrain the choice of appropriate action-objects. In LICAI, action-objects are relatively fine-grained physical responses (e.g., MOVE MOUSE CURSOR, DOUBLE CLICK). Thus, the responses of users are truly "constructed," as opposed to recalled as an organized schema.

The LICAI model was specifically designed to model the exploration of computer GUIs. Kitajima and his colleagues have since applied concepts of LICAI to deliberate searches of the Web for specific information. CoLiDeS represents further development of the C-I/LICAI tradition with two notable improvements (Byrne, 2002):

1. An explicit model of attention to guide the user toward acting on a particular screen object
2. A model of "informational scent-following" based on the semantic similarity of Web page labels and content.

c. LSA

LSA (Latent Semantic Analysis) is a theory of how people learn word meanings from patterns of actual usage (Landauer, 1998; Landauer and Dumais, 1997; Landauer, Foltz, and Laham, 1998). The theory is based on mathematical and statistical techniques that calculate the interrelationships among passages and items within passages for various print-based corpora, such as encyclopedias, newspaper text, or more specialized sources (e.g., psychology textbooks). The analyses reduce these huge databases to 100–500 latent dimensions. Individual items or passages are represented as vectors in this high dimensional space. The cosine between vectors measures the relatedness between items or passages, and vector length measures the importance of the item relative to the training corpora.

LSA has been used to simulate various tasks that are heavily dependent on word meanings. For instance, LSA has been shown to predict synonym judgments between pairs of words; simulate results from lexical priming experiments; and mimic synonym, antonym, singular-plural, and compound-compound word relationships (Landauer, Foltz, and Laham, 1998). Perhaps one of LSA's more interesting practical applications is in essay grading (Landauer et al., 1997). Most relevant to this review, however, are the applications of LSA to C-I Theory and to CoLiDeS (Kitajima, Blackmon, and Polson, 2000).

In C-I Theory, the strength of associations between propositional nodes is a function of the number of shared arguments. An implication is that the coherence of a passage can be measured by the average strength of associations among individual items. The problem was that the process of identifying all propositions and their arguments had to be done by hand. LSA can be used to identify potentially associated items and to provide a surrogate measure of passage coherence. Kintsch (2002) also showed how LSA and C-I Theory combine to explain the effects of context on predicates or simple noun-verb sentences.

To the extent that LICAI incorporates C-I, it could use LSA to quantify relations among propositions. However, LSA has a more explicit role in CoLiDeS, where it provides the measure of relatedness between the representation of a user's goal and screen objects, thereby operationalizing the concept of information scent (Kitajima, Blackmon, and Polson, 2000). It is also used as a Web site design aid to either confirm or identify appropriate headings and/or link labels for Web pages according to the correspondence between these elements and their corresponding passages (Blackmon et al., 2002).

3. Cognitive/Behavioral Functions Represented

Although the process of comprehension appeals to perceptual concepts, specific perceptual processes are not represented in these models. Rather, the model requires users to "pre-process" input in the form of propositions.

On the other hand, the CoLiDeS model does provide attentional mechanisms that segment or parse the display into elements based on perceptual features, such as highlighted or outlined areas, and then focus on parts of the display that relate most directly to the user's task goals (Blackmon et al., 2002). Similarly, psychomotor actions are primitively represented. Actions are merely selected—not specified in detail.

C-I and LICAI/CoLiDeS models assume the traditional distinction between LTM as the locus of permanently stored knowledge and WM as the currently activated subset of LTM elements that comprise one's momentary focus of attention. Ericsson and Kintsch (1995) point out that WM is traditionally conceived as being limited and temporary and that transfer to and from LTM is slow and error-prone. This characterization does not explain expert performance where skilled actors have virtually unlimited access to LTM and are able to encode large amounts of information quickly into LTM. To account for expert performance, which includes the comprehension skills of ordinary readers, Ericsson and Kintsch specify conditions under which WM provides performers relatively unconstrained access to LTM through well-structured retrieval cues. Because this capacity, called long-term working memory (LTWM), requires specific retrieval and storage structures built up through experience, LTWM—unlike its short-term counterpart—is not a generalizable capacity. In terms of C-I Theory, WM corresponds to the limited number of nodes to which a performer can attend during any one C-I cycle. In expert performance, those limited nodes can be connected to a complex structure of additional nodes in LTM that become activated during performance. This concept of LTM represents the scope and complexity of human knowledge, which can be modeled by LSA (Kintsch, Patel, and Ericsson, 1999). According to this concept, items in WM automatically generate an LTWM for associated items in its immediate semantic neighborhood as defined by LSA.

With regard to higher cognitive functions, Kitajima and Polson (1997) characterize C-I Theory as incomplete because it does not include specific mechanisms for learning and problem solving. Decision-making is also not explicitly represented, although C-I and LSA do model simple choice behavior. Also, even though C-I and

related models represent individual competence and performance, no provisions are available for modeling social interactions.

4. Applications

C-I has been used to model text comprehension and action planning. Although the C-I model has been used primarily in academic research, the associated models (LICAI/CoLiDeS and LSA) have been oriented toward practical applications. As a model of HCI, LICAI has been used to help design GUIs. Similarly, CoLiDeS has been used to design Web pages. Details on applications of LICAI and CoLiDeS are available on Muneo Kitijama's Web page at Japan's National Institute of Advanced Industrial Science and Technology: http://staff.aist.go.jp/kitajima.muneo/index.html.

LSA also has several applications as a tool for C-I and CoLiDeS and other semantically based models of cognition. LSA can be used as a general performance aid for writers and, with additional tools, can be used to grade essays automatically. Details on these applications can be found at the Web site for Knowledge Analysis Technologies (http://www.knowledge-technologies.com), the company that provides services based on LSA technology.

No evidence exists that any of these models has interacted with other cognitive models or with a military simulation.

5. Technical Considerations

a. Input and Input Aids

In contrast to most symbolic cognitive models, C-I and related representations do not require the user to input or otherwise specify a detailed rule or knowledge base. For some applications, the LSA database provides the contents of knowledge.

C-I models do, however, require the user to translate the simulated performer's input into a form the model can use. For C-I and LICAI, this input must be in the form of propositions. This process is not automated, but procedural manuals for propositional analysis are available to guide the modeler (e.g., Bovair and Kieras, 1984; Turner and Greene, 1977). CoLiDeS does not require propositions as input, but it does require the modeler to segment the display(s) into meaningful regions and labels that serve much the same purpose.

For LICAI and CoLiDeS, the modeler must also encode user goals. User goals for LICAI are input as informal propositions, whereas goals for CoLiDeS can be expressed in natural language (NL).

b. Model Output and Analysis Tools

Because the LICAI and CoLiDeS models are based on the process of comprehension, their primary output is the meaning constructed from input and constrained by knowledge. Generating behavior from this process requires the addition of production rules (action-object pairs). LICAI and CoLiDeS have been used to predict the simple choice behavior of computer users, and LICAI has even been used to predict errors in display-based HCI (Kitajima and Polson, 1995). No specialized tools are available for analyzing model outputs.

c. Computer Language and Interfaces

A computer implementation of C-I Theory was designed by Kintsch and Jon Roberts, an ICS computer programmer. This version was designed for Mac OS 9.1, but it may work for more recent versions. (No MS Windows version is currently available.) The software and an accompanying user's manual (Mross and Roberts, 1992) can be downloaded free from the following ICS file transfer protocol (FTP) Web site: ftp://psych.colorado.edu/pub/CI-Model.hqx.

The LICAI model has been implemented using Mathematica programs (Kitajima and Polson, 1998; Kitajima, Soto, and Polson, 1998). These models typically include algorithms to simulate the model's computationally intensive processes, such as spreading activation. However, large parts of the models are not computer implemented. For instance, the cycling process is only semi-automated. The choice of exactly what chunks are carried over to the next cycle is specified by the modeler. Again, Kintsch and Vipond (1978) suggest a "leading edge" strategy, which biases selection toward superordinate and recent propositions.

CoLiDeS is much more qualitative in nature and is designed to be run interactively with a Web page designer. Marilyn H. Blackmon's Web page at the University of Colorado's ICS—http://psych.colorado.edu/~blackmon/CWW.html—contains procedural guidance on using CoLiDeS. One of Muneo Kitajima's Web pages—http://staff.aist.go.jp/kitajima.muneo/CoLiDeS_Demo.html—contains a demonstration of the model's use.

Tools for using LSA and access to some LSA databases are directly available from the LSA Web site: http://lsa.colorado.edu.

6. Evaluation

One of the greatest disadvantages of symbolic models is that most—if not all—of their knowledge base must be identified before model execution. However, that is not true for these models (C-I Theory, LICAI, and LSA). They are especially applicable to tasks that have no clear goals. The model comprehension processes can be used to determine user goals. It is also good at describing search-and-error-prone behavior in HCI.

Ritter, Shadbolt, et al. (2002) also pointed out the following: Because of the research lineage of C-I, these models provide a relatively high-fidelity representation of text comprehension. However, many of the other types of basic cognitive processes (e.g., perception, learning, problem solving) are not represented in these models.

Although aspects of C-I and related models are computational, the models are not fully integrated enough to simulate human action in a dynamic environment. Consequently, considerable development would be needed to convert these models into a form that military simulations could use.

K. DISTRIBUTED COGNITION (DCOG)

1. Model Purpose and History of Development

Robert G. Eggleston, Kate McCreight, and Michael J. Young developed the Distributed Cognition (DCOG) model at the Air Force Research Laboratory (ARFL) Human Effectiveness Directorate at Wright Patterson Air Force Base, Ohio. Eggleston, Young, and McCreight (2000) explained that they use the phrase "distributed cognition" in two senses: Agent cognition is *not* under the control of a single executive, and individual agents respond directly to environment, effectively distributing cognition through environmental objects. R. G. Eggleston described the model's purpose and aim as follows:

> DCOG is an emerging computational architecture that supports complex and adaptive behavior. Its aim is to support the modeling of expert behavior in complex work in a manner that allows multiple strategies or methods by which an agent achieves expert performance (personal communication, October 9, 2002).

DCOG is a computer representation of the concepts of cognitive engineering developed by Jens Rasmussen and his colleagues at the Danish Atomic Energy Commission Research Establishment Risø. The primary impetus for DCOG was for it to serve as an HBR in the Agent-based Modeling and Behavior Representation (AMBR) model comparison project, which was sponsored by AFRL and ONR, with some limited funding by DMSO. The purpose of the AMBR project was to advance the state of the art in cognitive modeling by demonstrating and comparing the ability of four computer-based HBRs (DCOG, Soar, COGNET, and ACT-R) to emulate human behavior on a computer-controlled ATC task. The initial developers workshop was held in October 1999. The research was conducted in three rounds. Gluck and Pew reported the results from Round 3 at the Cognitive Science Society Meeting in July 2002 (Gluck and Pew, 2002).

Although DCOG was designed with a particular requirement in mind, Eggleston and his colleagues plan to continue to develop DCOG into a more general and robust model of human capabilities and limitations (R. G. Eggleston, personal communication, October 9, 2000). Their immediate plans are to respond to the newest AMBR challenge problem, which will require a more formal architecture of memory. Long-term plans are to apply DCOG to other work domains.

2. Principal Metaphors and Assumptions

The work domain in DCOG, which is adapted from Rasmussen's (1983) framework, is described by a 2-D matrix called the Abstraction-Decomposition Space (ADS) (Eggleston, Young, and McCreight, 2000). The abstraction dimension classifies work knowledge with respect to its focus on the ends-means distinction, with knowledge ranging from the most general functional meanings to the specific aspects of physical processes and material forms. The decomposition dimension classifies knowledge with respect to the whole-part relationship, which ranges from the total system on the one end to the individual components of the system on the other end.

Using verbal protocol analyses, Eggleston, Young, and McCreight (2001) showed that a trace of cognitive elements reveals subtle shifts in cognitive strategies used to solve complex real-world problems. These researchers contrasted the traces of engineers and electronic technicians while these engineers/electronic technicians undertook the same troubleshooting task. Although both performers start at the most abstract areas in the space and end at the most concrete areas, these researchers showed that technicians

employ much less reasoning about the functional characteristics of the system than do engineers.

Eggleston, Young, and McCreight (2001) also cited research indicating that individuals are able to change their path dynamically through the ADS in reaction to the environment. For instance, under low cognitive workload, individuals can choose to explore the abstract aspects of the task. Under higher workload conditions, the same individuals can stick to the more concrete aspects of the ADS in an effort to minimize cognitive storage and processing demands. The preference for one or the other strategy is dependent not only on environmental conditions, but also the performers' preferred level of arousal—that is, some individuals may seek more difficult solutions to increase arousal, while others choose less difficult solutions to decrease their arousal level. The DCOG model was specifically designed to model shifts in strategy caused by environmental changes and/or individual differences in experience and disposition.

The ADS space for the ATC task in the AMBR project was depicted as a goal hierarchy (Eggleston, Young, and McCreight, 2001). At the highest level of abstraction is the functional purpose of the ATC task: to provide coordinated use of airspace. This overall purpose is parsed into two abstraction functions or goals: avoid preventable delays and prevent accidents. Each of these two goals is parsed further into a set of lower-level behavioral strategies and actions. The links among elements in this hierarchy can be varied in strength to model the different trajectories that workers can take through the space.

In accord with the Rasmussen taxonomy, DCOG emulates differences in the level of information processing as a function of the information observed in the environment (Eggleston, Young, and McCreight, 2000; Rasmussen, 1983). The lowest level is *skill-based* processing, which is triggered by environmental stimuli perceived as *signals*. Signals are continuous quantitative indications of environmental states that do have not symbolic meaning. Because signals do not require cognitive processing, they serve to connect the worker directly with the environment. The next level is *rule-based* processing, which is initiated by *signs*. Signs are stimuli that provide information about the states of the environment and are directly associated with a particular object or action. The highest level is *knowledge-based* processing, which is governed by perceived *symbols*. In contrast to the signals or signs, *symbols* require deliberate cognitive processing of meaningful information. The worker's current proficiency and his or her present task determines whether a stimulus is perceived as a signal, sign, or symbol. For instance, a novice

performer may perceive a stimulus as a symbol that requires considerable knowledge-based processing to interpret, while an expert performer may perceive the same stimulus as a sign that is directly associated with a particular action. In DCOG, such experience differences are modeled as variations in knowledge structures and cognitive strategies.

3. Cognitive/Behavioral Functions Represented

DCOG does not model psychomotor phenomena. However, perception and attention are modeled though schema-type knowledge structures that directly recognize meaningful situations (e.g., aircraft moving toward an area border). In addition, two types of eye movements are modeled: visual scanning and directed gaze. Eye movements in visual scanning are for identifying "areas of tension" in the display, whereas a directed gaze is used to follow specific targets while actively controlling or handing off the aircraft.

LTM is modeled as storage for procedural information that is organized by schemata. WM is modeled as the activation of one or more of these schemata in memory.

DCOG does not model the acquisition of memory contents (i.e., learning). However, the latest AMBR requirement stipulated that DCOG model the *effects* of learning a secondary task (concept learning) that was embedded in the primary task [ATC (air traffic control)]: Subjects had to learn to make correct decisions to accept or reject altitude change requests, based on three bi-variate properties of the aircraft (percent fuel remaining, aircraft size, and turbulence level) (Diller and Tenney, 2002).

Reactive decision-making is implicitly modeled in the selective activation of schemata. DCOG does not model more deliberative forms of decision-making or problem-solving behavior.

A unique aspect of DCOG is its ability to model workload effects. In DCOG, workload is defined with reference to the number of actionable schemata in WM (Eggleston, Young, and McCreight, 2001). If workload is low (three schemata or fewer), the simulated performer scans areas of tension and updates aircraft information, waiting to process the next aircraft that nears the transition border. If workload is high (four or more schemata), the simulated performer processes the aircraft closest to the border without the scanning or updating.

DCOG is a model of individual performance. As such, it provides no capacity to model social or collective behaviors.

4. Applications

So far, DCOG has been applied to only two ATC tasks: transferring aircraft from one sector to another and responding to requests for speed increases. Developers are seeking new applications, but none have been identified (R. G. Eggleston, personal communication, October 11, 2002).

5. Technical Considerations

a. Input and Input Aids

Input to the DCOG model includes a complete and detailed specification of task strategies and the conditions under which each is applicable. The source code for DCOG is not in releasable form, and no documentation or aids are currently available to support development of models (R. G. Eggleston, personal communication, October 11, 2002).

b. Model Output and Analysis Tools

As stipulated by the original AMBR challenge (Tenney and Spector, 2001), DCOG generated the following output variables: response times, workload ratings, and a composite penalty score designed to measure overall controller performance. In the latest version of the challenge (AMBR III), a requirement was added: to measure the number of errors committed on the concept-learning task as a function of practice.

c. Computer Language and Interfaces

The original version of DCOG was implemented in Allegro Common LISP. The latest version is implemented in Java 2 v1.4, and the developers plan to stay current with Java updates (R. G. Eggleston, personal communication, October 11, 2002). As stipulated by the AMBR challenge, DCOG was interfaced with the ATC simulation implemented in Distributed Operator Model Architecture (D-OMAR) via a High-Level Architecture (HLA) Run-Time Interface (RTI).

6. Evaluation

The unique feature of DCOG is its ability to model individual differences and workload effects in an explicit fashion. On the other hand, the utility of DCOG for general application is seriously limited by the fact that the model has been developed for a single job task and the software has not been documented and is not in a releasable form.

L. EXECUTIVE PROCESS/INTERACTIVE CONTROL (EPIC)

1. Model Purpose and History of Development

The purpose of the Executive Process/Interactive Control (EPIC) model was to describe in detail the peripheral cognitive processes—that is, the perceptual and the psychomotor processes. The developers, David E. Kieras and David E. Meyer at the University of Michigan, specifically acknowledged that the MHP (Model Human Processor) developed earlier by Card, Moran, and Newell (1983) served as the basic framework within which they devised the EPIC model. However, in contrast to MHP, which is a practical model of cognitive processes relevant to HCI, Kieras and Meyer (1995) regard EPIC as having broader and more scientific goals because it represents "…a larger scientific endeavor to represent important theoretical concepts of human intelligence or abilities" (p. 1).

EPIC is currently under active development by the Brain, Cognition, and Action Laboratory at the University of Michigan. The laboratory is directed by David Meyer and is sponsored by ONR. More information is available at the David Meyer's EPIC Web site: http://www.umich.edu/~bcalab/epic.html. The Principal Investigator (PI) for the EPIC project is David Kieras: http://www.eecs.umich.edu/~kieras/epic.html.

EPIC is currently being used to make predictions of outcomes from laboratory experiments, but it is also touted as serving an engineering function in modeling human-system interaction and designing appropriate interfaces.

2. Principal Metaphors and Assumptions

An important impetus for EPIC was the developers' sense that the current (i.e., circa 1995) HBR models represented individual humans as a pure cognitive system or "disembodied intelligence" that directly perceives and acts on its environment (Kieras and Meyer, 1995). EPIC was designed specifically to model those input/output (I/O) details of perception and motor action. For instance, visual processing is modeled as two separate processes: sensation and perception, with the latter process controlling ocular motor movement. Likewise, motor processing is modeled by a detailed three-phase process. The movement is

1. Prepared by generating a list of movement features
2. Initiated after all features are processed

3. Executed by making the desired mechanical motions.

The EPIC architecture is comprised of a set of interconnected processors that operate simultaneously and in parallel. The processors are standard ruled-based constructs (i.e., production systems) under the control of an executive process. Although the executive process assumes a central role in EPIC, it is modeled as a set of production rules no different from other rules: The executive works by enabling and disabling other productions or by controlling the sensory/motor peripherals directly. In other words, executive control is not modeled as a separate cognitive component but rather as an integral part of the cognitive system.

3. Cognitive/Behavioral Functions Represented

Clearly, one of EPIC's strengths is that it provides detailed models of sensory and motor processes. Visual processing is modeled as two separate processes: sensation and perception. Likewise, motor processing is modeled by a detailed three-phase process that controls the preparation, initiation, and execution of movements.

The model includes several distinct sites for memory storage, including a long-term store of declarative knowledge about performing the tasks in question, a procedural memory of compiled productions, and a WM that contains symbolic information for testing and applying production rules in procedural memory. In contrast to most other rule-based models, central memory stores in EPIC are *not* limited in either capacity or retrieval processing; rather, the observed limitations in human information processing are modeled as limitations in peripheral (i.e., perceptual or motor) systems.

The EPIC model provides no model of the learning processes or of problem solving. Because it is a model of individual performance, it has no capability to model collective behavior. A reactive sort of decision-making is emulated via the production system rules, and attention is controlled through the executive process, which is a subset of those rules.

4. Applications

EPIC, which was developed primarily to model HCI, was designed to model simple, dual-task situations. It has been applied primarily to laboratory tasks that serve as analogues of those situations. Meyer and Kieras (1997a; 1997b) describe some of EPIC's laboratory research applications. In the practical arena, EPIC has been used to model telephone operator call-completion tasks.

5. Technical Considerations

a. Input and Input Aids

Some of the knowledge base and performance parameters are predefined in EPIC. The user must completely specify the task environment (i.e., the objects with which the simulated human is to interact) and the knowledge specific to the task. The production system knowledge base is input through the PPS (Parsimonious Production System) interpreter. Kieras and Meyer (1998) provide some technical guidance on the EPIC architecture, although they explicitly caution that this document does not provide the details necessary to build EPIC models.

Many of EPIC's features of are now embodied in the GLEAN model (Kieras, 1999). Rather than creating and modifying complex databases from PPS productions, GLEAN directly imports results from GOMS task analyses as coded in GOMS Language (GOMSL). The discussion of CCT in Section II.G provides more details about GLEAN and GOMSL.

b. Model Output and Analysis Tools

The fundamental output for EPIC is the trace of model methods (steps) executed. If the EPIC model is implemented in GLEAN, the software system includes additional statistics such as frequency, average time, and total execution time of each operator and method. The software also tracks other indicators, such as the contents of WM and the state of the simulated device.

c. Computer Language and Interfaces

The standalone EPIC model is designed for Apple Power PC Macintosh platforms using Common LISP, Version 4.3. EPIC source code and installation instructions are available at Web site ftp://ftp.eecs.umich.edu/people/kieras/EPIC/EPIC_distribution. Potential users are cautioned to contact David Kieras (kieras@eecs.umich.edu) before trying to use the software.

The cognitive processor, which contains and controls WM and the production rule interpreter, has an update rate of 50 msec. The other components also have associated processing times, but they can be set by the user.

EPIC has been successfully incorporated into Soar and ACT-R.

6. Evaluation

Of the macro models, EPIC is probably the most constrained (being focused on the details of input and output). This characteristic makes it a good research tool for making precise predictions about the order and timing of responses. At the same time, this characteristic limits its practical import. Perhaps its greatest benefit to simulation is its ability to be used in hybrid systems to enhance the perceptualmotor aspects of other models, such as Soar and ACT-R.

M. HUMAN OPERATOR SIMULATOR (HOS)

1. Model Purpose and History of Development

The purpose of the Human Operator Simulator (HOS) is to provide a model of human capabilities and limitations to support the design of human-machine systems. HOS is an example of a Siegel-Wolf discrete event model that dates to the late 1950s (Siegel and Wolf, 1962; 1969). The fundamental assumption of these models is that the dynamics of human performance can be derived from the sequence and timing of discrete subtasks. The organization of subtasks is described by a network metaphor; thus, Siegel-Wolf models are sometimes referred to as "task network" simulations.[12] The innovation of HOS is that it integrates the concept of a task network with "micro-models" of human perceptualmotor and cognitive performance (Zachary, Campbell, et al., 2001).

HOS has been updated several times since its initiation. The initial version was developed by R. J. Wherry, Jr. (1976) for the U.S. Navy. The last standalone version of the model (HOS V) was developed by Glenn, Schwartz, and Ross (1992). However, much of the logic and functions of HOS have been incorporated into two models, which are under active development: the present version of COGNET model (see Section II.F), and the Integrated Performance Modeling Environment (IPME) of Micro SAINT (see Section II.O).

2. Principal Metaphors and Assumptions

One approach to describing HOS is to focus on assumptions. Perhaps the most central assumption of HOS is that human performance is described by a network of

---

[12] The SAINT (Systems Analysis of Integrated Network of Tasks) model, the predecessor of Micro SAINT (described in Section II.O), is another descendent of these Siegel-Wolf concepts.

discrete subtasks. Another general assumption is that the time to complete a task is calculated as the sum of times required to execute the component subtasks and required processes—that is, the model assumes additivity of components. In addition to these general assumptions, Pew and Mavor (1998) identified the following three specific assumptions that constrain HOS models:

1. The human operator has only a single channel of attention. Time-sharing is accomplished through rapid attention switching.
2. Operator tasks are all highly proceduralized and predictable. HOS can represent simple decisions but not complex problem solving involving open-ended tasks.
3. The human operator does not commit errors during task performance.

Harris, Iavecchia, and Dick (1989) describe the HOS software as being based on the creation of interfaces among three major simulation components:

1. **Simulation objects.** Simulation objects represent the configuration of displays and controls. Objects are sorted into categories (e.g., dials, digital displays) and defined by a list of attributes (e.g., size, color, position).
2. **A task network.** A task network defines the procedures used to operate the crewstation. Model actions are represented as a verb-object pair (e.g., turn on switch). These actions can be embedded in rules, which specify the conditions under which the action should be taken.
3. **Micro-models.** Micro-models describe how the human interacts with the displays and controls. These micro-models are software modules that calculate the length of time required to complete each subtask based on component processes that underlie task performance (e.g., perception, information processing, motor response).

3. Cognitive/Behavioral Functions Represented

The time to perceive information from the environment is explicitly modeled from estimates of the time to touch or to fixate visually and then mentally derive information from visual displays. Similarly, response times are modeled by the time required to perform an action using a particular body part. If that part is busy, the model simulates the time required to choose an alternative method (e.g., swap hands) and execute the action.

Attention is explicitly modeled as a single-channel process in which only one subtask is executed at a time. HOS maintains a list of subtasks to be completed and chooses

among them based on a priority assigned by the user or calculated from other factors, such as the time it has been attended to and the idle time since initiation. Changes to priority or environmental stimuli may lead to the interruption of the execution of one subtask and the initiation of another. The processing of an interrupted subtask is resumed at the point of interruption. Rapid attention switching is assumed because such interruptions and resumptions consume no processing overheads.

The network of subtasks implies some form of permanent (i.e., long-term) memory; however, LTM effects (e.g., interference or organization) are not explicitly modeled in HOS. In contrast, STM is explicitly modeled as a decay function that degrades the probability of retrieving an estimated value or previous state as a function of time. HOS includes no representation of learning.

If-then type of statements are used to simulate simple reactive decision-making. HOS adds another layer of fidelity by calculating decision time based on the number of alternatives and the complexity of the decision. However, other higher order cognitive processes (i.e., problem solving and comprehension) are not modeled.

4. Applications

In general, HOS has been used to simulate performance of humans interacting with some sort of display. The initial application of HOS was for describing person-machine interactions in an air antisubmarine task for the Navy (Wherry, 1976). Glenn and Doane (1981) also used HOS to predict eye-scan behavior in NASA's Terminal Configured Vehicle (an experimental Boeing 737 airliner).

5. Technical Considerations

a. Input and Input aids

The user must input all environmental objects (e.g., displays and controls) and a complete task and subtask hierarchy that specifies sequential dependencies among subtasks. In addition, subtask execution times must be specified. Based on extensive reviews of the literature, many of those parameters have default values; however, the user can change those values as needed. Initial versions of HOS were deterministic because users entered only single values for parameters. Later versions allowed users to specify distributions of input values, which are used to generate Monte Carlo runs of the simulation. The latest version of HOS (Version V) includes editors, libraries, and data files to support

the development and use of HOS models. HOS V also includes a graphic aid for defining the task network.

b. Model Output and Analysis Tools

The principal output for HOS is a list of messages about tasks that were executed during simulation runs. Each message contains the time, the type of behavior, and the object (display, control, function, procedure) to which the behavior was directed. An HOS component program, the Human Operator Data Analyzer/Collator (HODAC), uses these output data to construct time lines and to accumulate statistics on time spent by body part, actions, procedures, steps within procedures, and interactions among devices. If the model is run in a Monte Carlo mode, distributions of results are provided.

c. Computer Language and Interfaces

Earlier versions of HOS were written in FORTRAN. The latest version (HOS V) is written in C to run on MS DOS. No known instances exist to indicate that HOS has been interfaced with a military simulation.

6. Evaluation

It is reported that HOS is relatively easy to use and is based on relatively well-validated psychological data and models. At one time, HOS was the most popular man-machine model for human-factors applications (Green, 1999). On the other hand, the output from HOS is limited to response times, which has narrow relevance to military simulations. The ultimate problem with HOS, however, is that it is no longer supported as a standalone model. The current interest in HOS stems from its historical significance as a computer-based human behavior model and from aspects of HOS that have survived as components to two HBR models—COGNET and Micro SAINT—that are under active development and have been interfaced with military simulations.

N. MAN-MACHINE INTEGRATED DESIGN AND ANALYSIS SYSTEM (MIDAS)

1. Model Purpose and History of Development

The U.S. Army and NASA, supported by Sterling Software Incorporated,[13] began development of the Man-Machine Integrated Design and Analysis System (MIDAS) in 1983 as part of the Army Aircrew/Aircraft Integration (A3I) Program at the NASA ARC. Jim Hartzell, Barry Smith, and Kevin Corker produced the initial version of MIDAS in 1986 (Corker, 2001). According to Corker and Smith (1993), the purpose of MIDAS is

> …to revise the system design process in order to place more accurate information into the hands of the designers early in the process of human engineering design so that the impact and cost of changes are minimal. It is also intended to identify and model human/automation interactions with flexible representations of human-machine function (first page of article).

Although the primary purpose of MIDAS is to provide an engineering aid, it is also regarded as an exploratory development project whose purpose is to advance the state of human performance modeling. As such, MIDAS has been revised continuously by the NASA ARC team since its initiation. In 1996, a major rearchitecturing project was begun by NASA ARC [Sherman Tyler (Sterling Software) and Jay Shively (government project officer)]. MIDAS v2.0 was released in 2001 but is still in the Beta release mode (Hart et al., 2001). The NASA ARC work on MIDAS is described at its Web site: http://caffeine.arc.nasa.gov/midas/index.html.

One of the MIDAS principals, Kevin Corker, left the A3I program in 1999 to head San Jose State University's Human Automation Integration Laboratory (HAIL). However, he and his colleagues at HAIL remain involved with the A3I program at NASA ARC in the development of the original MIDAS model (now called "Core MIDAS"), and HAIL has taken the lead in developing an elaboration of MIDAS, which has been named "Air MIDAS." Information on HAIL's involvement in both projects is available at http://www.engr.sjsu.edu/hfe/hail.

---

[13] Sterling Software was acquired by Computer Associates in March 2000.

2. Principal Metaphors and Assumptions

MIDAS developers have described their model as one that is derived from first principles—that is, MIDAS modules are explicitly designed to model human capabilities and limitations. At the same time, MIDAS is not designed to be a unified theory of cognition; rather, it is intended to be first and foremost a design aid (Gore and Corker, 1999). According to Pew and Mavor (1998), the most fundamental assumption made by MIDAS is that "…the human operator can perform multiple, concurrent tasks, subject to available, perceptual, cognitive, and motor resources" (p. 75). These assumptions are completely consistent with the object-oriented agent structure of the simulation software.

According to Tyler et al. (1998), the most recent revision of the MIDAS architecture is divided into five high-level components:

1. The domain model that supports the components necessary to run the simulation
2. A graphics system to enable users to visualize the simulation
3. An interface to allow users to input model specifications
4. The simulation system for controlling the simulation and collecting data
5. The results analysis system for analyzing simulation data after these data have been collected.

The domain model is further decomposed into the *environment*, which encompasses the crewstation; the *vehicle*, which contains the crewstation; the *crewstation* itself; and the *human operators*, which comprise the crew.

One of the principal purposes of the MIDAS rearchitecture was to align the human operator model more closely with typical models of human information processing (Tyler et al., 1998). According to Hart et al. (2001), the human operator model comprises six components: sensory input, memory, decision-making, attention, SA, and output behavior.

a. Sensory Input

Visual and auditory inputs are processed by separate models of sensation and perception. Sensory operators query the visual or auditory scene to determine whether they contain perceivable objects. Perception, on the other hand, depends on the objects' associated attributes and information about surroundings (e.g., ambient light and noise). The visual model distinguishes between foveal vision, which simulates fixation and attention

to specific objects, and peripheral vision, which emulates the human's ability to be distracted by visual stimuli to which he/she is not attending. Auditory processing is modeled in two stages: detection and comprehension. Partial comprehension is not allowed, and interruptions to auditory input result in losing the entire message. MIDAS' visual processing of objects outside of the crew station proceeds in three stages: detection, recognition, and identification. Inside the crewstation, identification is automatic because the crewmembers are assumed to have an adequate mental representation of the crewstation equipment.

b. Memory

Memory is divided into declarative and procedural components. LTM for declarative information is represented by a database of assertions ("beliefs"), whereas WM includes current beliefs that define the present context. Beliefs are encoded as symbolic expressions describing properties of objects.

Procedural knowledge is represented by a set of primitive procedures. WM refers to the subset of procedures under active processing by the procedure interpreter and is limited in capacity.

A scheduler controls the flow from WM to LTM. This transfer is limited such that if some threshold value is exceeded, attributes of the activity in WM are forgotten but not the memory tag associated with the activity.

c. Decision-Making

Reactive decision-making processes are encoded as procedures using the high-level scripting language called Operator Procedure Language (OPL). As a programming language, OPL takes input (arguments) and invokes other procedures. OPL can model simple procedures and more complex behaviors, such as selecting between alternatives, repetition, passively monitoring for a perceived condition, and performing concurrent tasks.

d. Attention

This module, which is based on Wicken's Multiple Resource Theory (MRT) of attention, monitors an account of six different "channels": visual input, auditory input, spatial cognitive processing, verbal cognitive processing, motor output, and voice output. Before initiating an action, the relevant resources are secured. If sufficient resources are

not available, task performance is systematically degraded. This module is also consulted to estimate workload within each of the channels.

e. SA (Situation Awareness)

Jay Shively developed the MIDAS model of SA. This model computes two quantities: actual SA vs. perceived SA. Actual SA is the portion of situational elements that the operator knows relative to the situation elements that he would know under ideal conditions. Perceived SA is similarly defined but does not include elements for which the operator has no knowledge.

f. Output Behavior

Behavioral output is composed of primitive motor actions. These actions have effects on equipment and on simulation objects. To visualize actions, MIDAS v2.0 includes two anthropometric models. Jack® is a model of a full body, which was developed by Norman Badler at the University of Pennsylvania and is currently marketed by Unigraphics Solutions, Inc. Because of licensing restrictions associated with Jack® and the computer processing requirements of running Jack®, MIDAS v2.0 also includes a simpler model that includes just the head and hands. This latter anthropometric model is the only one implemented in Air MIDAS.

3. Cognitive/Behavioral Functions Represented

Many of the details about cognitive and/or behavioral functions are intrinsic to the model and were therefore described in greater detail under "Principal Metaphors and Assumptions." The following summarizes those functions.

As described previously, MIDAS includes a symbolic simulation of visual and auditory perception. Action primitives model motor processes and their effects. Input and output processes, as well as cognitive processes, are under the control of an attention process based on Wicken's MRT.

MIDAS models the experienced performer and thus has no provision for learning. On the other hand, the storage and retrieval functions of LTM and WM are modeled. WM, which is viewed as a subset of LTM, is modeled as a capacity-limited process.

Reactive decisions, even those involving complex dependencies, can be modeled using OPL. However, more complex cognitive processes (e.g., problem solving) cannot be modeled without substantial development.

One of the intriguing aspects of MIDAS is that it has the capability to model multiple crewmembers and crewstations. Thus, it is able to model the interactions among crewmembers.

4. Applications

The first application of MIDAS in 1985 was to model a military mission performed by an Army Cobra AH-1 attack helicopter. Since then, MIDAS has been used to simulate a varied set of tasks and vehicles, including

- Use of a proposed unmanned underwater vehicle for seeking out and destroying mines on the ocean floor (U.S. Navy)
- Design of high-speed wireless communication and navigation systems for emergency response vehicles and 911 dispatch stations (Communications Research Company)
- Automation options for the next-generation nuclear power plant (Westinghouse)
- Development of an improved Mission Oriented Protection Posture (MOPP) ensemble for helicopter aircrews (U.S. Army)
- Evaluation of a civilian version of the Marine V-22 Osprey tilt-rotor vertical take-off and landing (VTOL) aircraft (NASA)
- Advanced air traffic technologies for commercial transport operations (NASA)
- Design of an upgrade to the cockpit instrumentation in the Space Shuttle (NASA).

5. Technical Considerations

a. Input and Input Aids

Using flight tasks as a model, the user must specify three sets of inputs:

1. The crewstation design (cockpit geometry, display/control layout, and equipment functionality)
2. Details concerning the mission and its context (task lists, planned operator activities, flight profiles, waypoints, scenarios objects)
3. Human operator characteristics (cognitive attributes and physical/motor attributes).

In MIDAS v1.0, users also had to specify all goals and procedures that operators use to reach those goals. With the advent of v2.0, users can specify more abstract behaviors, the details of which are then filled in by the library of primitive actions.

Input is aided by a GUI. This interface is organized into a set of hierarchical screens or editors that can be navigated by a tabbed file deck metaphor. The interface provides several different views of the simulation for editing and running user models.

b. Model Output and Analysis Tools

MIDAS provides two types of outputs. The first type of output is the results obtained interactively during MIDAS runs. Interactive analyses are supported by the animated anthropometric model and other animation views of the model in action (e.g., operator's view, wingman's view, "bird's eye" view). The user can also interactively view data displays to view specific changes in model states (e.g., workload). These interactive outputs support traditional human factors analyses, including usability standards such as MIL-STD-1472, *Design Criteria Standard. Human Engineering*.

The second type of output refers to post-run statistical analyses of the results from simulation runs. These analyses, accessed through the user interface, include operational time lines, information flow during performance, and summaries of mission effectiveness and other measures (e.g., workload and SA).

c. Computer Language and Interfaces

The original version of MIDAS (v1.0) was written in a combination of FORTRAN, LISP, C, and C++ to run on multiple computers. One of the goals of the MIDAS redesign was to reduce the large and unwieldy software computational requirements. According to Smith and Tyler (1997), the older model comprised 350,000 lines of code at one point, about half of which were devoted to dynamic anthropometry. The newer version of MIDAS (v2.0) is written completely in C++ and hosted on a single Silicon Graphics Workstation. Air MIDAS continues to use the older MIDAS combination of software, but plans to reintegrate the two versions of MIDAS into a C++ system are in progress.

MIDAS is a standalone constructive simulation system that has not yet been used to interact with other HBR models or military simulations. The model operates in scaled-time, as opposed to real-time. The simulation update interval is 100 msecs of simulated time.

6. Evaluation

The sheer size and extent of MIDAS is a strength and a weakness. It allows the user to model the detail of man-machine interactions and observe the results through sophisticated performance animations; however, it also makes creating and executing models difficult for the user. Also, while not impossible in principle, an effort to allow MIDAS to interact with other HBR models and simulations would require a significant development effort.

O. MICRO SYSTEMS ANALYSIS OF INTEGRATED NETWORK OF TASKS (Micro SAINT)

1. Model Purpose and History of Development

This HBR evolved from a model called SAINT (Systems Analysis of Integrated Network of Tasks), which was developed by the Air Force to simulate complex man-machine systems (Pritsker et al., 1974). Like HOS, SAINT evolved from the Siegel-Wolf concept of a task network—that is, that task performance can be reduced to performance on a hierarchy of discrete tasks and subtasks. The innovation of SAINT was to convert the modeling concepts of Siegel and Wolf (1962, 1969) into a general simulation tool.

K. Ronald Laughery, Jr. established Micro Analysis and Design (MA&D) in 1981 to develop a commercial version of SAINT for implementation on PCs.[14] The resulting product, marketed as "Micro SAINT," is a discrete event simulation environment that is designed for modeling complex processes. MA&D currently develops and markets two other products that use Micro SAINT as the simulation engine. The first is the Integrated Performance Modeling Environment (IPME), which incorporates the HOS micro-models to model the details of sensory input and motor output (see Section II.M). The second product is WinCrew, which models the experiences of individuals and/or crews during task performance to estimate workload. Micro SAINT, along with IPME and WinCrew, have been incorporated into the Army's Improved Performance Research Integration Tool (IMPRINT), which is a set of modeling tools that assess the interaction of soldier and system performance throughout the lifecycle of a military end item.

---

[14] As with HOS, SAINT was developed for a mainframe computer.

2. Principal Metaphors and Assumptions

Laughery and Corker (1997) characterize Micro SAINT as a *reductionist* model, in that it assumes that large, meaningful behavior acts (e.g., attack target) can be validly decomposed into successively smaller behavioral units (e.g., acquire target, engage target, and reengage target if necessary). The process of decomposition continues until an "elemental" level of analysis is reached, where analysts can validly provide estimates of performance. The resulting task network is a summary of the results from this hierarchical decomposition process. In addition to hierarchical relations, the network also specifies the branching logic and sequential dependencies that exist among task elements.

The task network is derived through a task analysis process. The resulting network comprises two types of elements: nodes and relationships. The nodes represent subtasks and are defined by the release conditions, which are the set of stimulus circumstances that must be met before the task is initiated; the time required to complete the subtask; and beginning and ending effects, which defined variables shared among nodes. These effects can also refer to external systems that can be modeled along with human performance to create closed loop human-machine systems. The relationships among nodes are defined by shared variables, which define the decision logic that determines alternate pathways through the task network. Three types of decisions are modeled:

1. Probabilistic, where the initiation of a particular subtask is dependent upon a random process
2. Tactical, where the initiation of a subtask depends upon a calculated value
3. Multiple, where more than one subtask is initiated at a time.

3. Cognitive/Behavioral Functions Represented

Micro SAINT is a simulation tool—not a cognitive model. As such, the basic tool does not include some cognitive functions. On the other hand, the generic nature of Micro SAINT's tools allows it to be modified to simulate some of these functions. To determine whether a function can be represented in Micro SAINT, it was assumed that the function has been implemented in either the basic Micro SAINT model, one of the Micro SAINT-related products (IPME or WinCrew), or published Micro SAINT algorithms.

Micro SAINT models basic perceptual phenomena, such as detection time and probabilities. More detailed models of perception are simulated by the IPME HOS engine, which is not included in the standard Micro SAINT package. Likewise, Micro

SAINT provides the basic simulation of the accuracy and timing of response outputs, but more detailed simulations are available only through the IPME HOS engine.

Micro SAINT does not explicitly model attention processes, although it does emulate serial and/or parallel processing. However, the CrewCut module (not a part of the standard package) models the effects of attentional processes on workload and stress.

The decision logic inherent in Micro SAINT enables a wide range of decision-making behaviors—from rapid recognition-primed decision-making to more deliberative processes. On the other hand, open-ended problem solving cannot be modeled by the software without a major development project.

As is common among task network models, the underlying task analysis focuses on expert task performance and does not account for learning, forgetting, or other error-inducing process.

4. Applications

Micro SAINT has been used to develop constructive or analytic models of performance on a wide variety of individual and collective tasks, including many military applications. Recent examples include

- Modeling cultural differences in the operation of the Integrated Air Defense System (IADS)
- Developing models of performance to be integrated into military weapons systems in the Combat Automation Requirements Testbed (CART)
- Simulating recognition-primed decision-making, a theory of natural decision-making under time stress.

5. Technical Considerations

a. Input and Input Aids

The fundamental input to Micro SAINT is a detailed task analysis, which includes the identification and classification of all subtasks and the interrelationships among them. The user also needs to provide values related to the time required to complete each subtask, including means, standard deviations, and distribution shapes.

Micro SAINT includes a full array of aids for inputting task data, including graphic editors for constructing task networks, developing task descriptions, and defining

task branching decision logic. Users can also refer to a library of predefined functions in creating their own functions.

b. Model Output and Analysis Tools

Micro SAINT output includes estimates of task completion times, accuracy of task outcomes, and measures of operator workload. It also has data collection and display modules for viewing results after computer runs and an animation viewer to observe output during the runs.

c. Computer Language and Interfaces

Micro SAINT and WinCrew are implemented in Microsoft Windows. IPME, in contrast, is a UNIX-based system written in C and C++. The present version is written for the Linux RedHat 6.2 (or better) operating system operating on Silicon Graphics IRIX 6.5 hardware.

Micro SAINT models have been interfaced with modular semi automated forces (ModSAF) to provide higher fidelity human behavior representations in that simulation (e.g., LaVine et al., 1999). In general, Micro SAINT communicates with other software applications through COM Services, an optional add-on module. This service facilitates the creation of middleware (using applications such as Basic, Visual C++, or HLA) between Micro SAINT and another application. This service allows users to start, stop, and continue the model by line command and to pass variable values into and out of Micro SAINT.

6. Evaluation

Micro SAINT is one of the most popular approaches for representing human behavior, partly because of MA&D's dedication to user support. Its popularity is also a result of its flexibility as a simulation tool. Because it is not wedded to a particular theoretical viewpoint, users have capitalized on its flexibility to simulate a variety of HBR phenomena using a range of theoretical mechanisms. For instance, Micro SAINT has been used to simulate decision-making using the recognition-primed decision-making model (Warwick et al., 2001). While such complex models and theory-driven models can be developed feasibly on Micro SAINT, they are sometimes more difficult and awkward to develop in a model that does not include some basic cognitive processes. For instance,

in the Warwick et al. (2001) study, a model of LTM, which is not included in Micro SAINT, had to be developed to simulate the memory-driven decision process.

P. OPERATOR MODEL ARCHITECTURE (OMAR)

1. Model Purpose and History of Development

Stephen Deutsch and his colleagues at Bolt, Beranek, and Newman (BBN) Corporation[15] developed the Operator Model ARchitecture (OMAR) for AFRL. The earliest technical reports on OMAR date to 1993 (e.g., Deutsch, Adams, et al., 1993; Deutsch, MacMillian, and Cramer, 1993), but the model developers trace OMAR's roots to work on distributed computing performed at BBN in the mid to late 1980s (Agha, 1986; Abrett, Burstein, and Deutsch, 1989). The original purpose of OMAR was to simulate the interactions of small numbers of human operators executing tasks and operating equipment in a complex environment. Such complex environments required operators to manage these actions in the face of frequent interruptions. ATC, to include the behavior of controllers and aircrews, was chosen as the archetypal environment for initial development of OMAR.

In 1998, OMAR was redesigned as a distributed system (Distributed OMAR or D-OMAR) to interact with a range of other applications/systems, namely other simulations or HBR models (Cramer, 1998). As a result, its purpose has been broadened to serve as a programming environment and testbed for developing human behavioral models. In that role, D-OMAR served as the central simulation in the AMBR project, sponsored by AFRL and DMSO. In the AMBR project, D-OMAR simulated the ATC environment within which other HBR models simulated human participants (Deutsch and Benyo, 2001). D-OMAR documentation and software can be downloaded from the BBN OMAR Web page: http://omar.bbn.com.

2. Principal Metaphors and Assumptions

One assumption of OMAR is that human behavior in a complex and interactive environment is proactive and reactive—that is, humans operate on the basis of some goal-oriented agenda but must also respond to frequent interruptions. Another characteristic of these environments is that tasks occur concurrently within and among multiple operators.

---

[15] The company is now known as "BBN Technologies" and is a subsidiary of Verizon Communications, Incorporated.

However, OMAR does not assume that these parallel and interactive activities and functions are under the control of a central executive. This aspect of OMAR is based on recent psychophysiological data indicating that behavior is controlled by a relatively small set of semi-independent brain centers (Deutsch, 1998; Deutsch and Adams, 1995). These data indicate that the smallest functional groupings comprise relatively large groups of neurons presumably controlling meaningful chunks of behavior. Deutsch (1998) interpreted these findings as indicating that modeling complex parallel processes could be accomplished entirely using symbolic mechanisms—that is, without using subsymbolic (i.e., associative) components to model individual neurons and their interconnections.

Based on these assumptions, OMAR models human behavior as interactions among independent computational agents. These agents can represent different people or different functions within a single person. An important aspect of OMAR is the absence of an executive process or scheduler that controls these parallel activities. The order of execution depends on the initiating conditions of procedures or the activation level of tasks. Thus, order emerges from the dynamics of task interactions that alter those activation levels. Thus, OMAR can specify explicit sequential dependencies among tasks while allowing some activities to occur in parallel.

3. Cognitive/Behavioral Functions Represented

For the ATC task, OMAR emulates audition and vision processing. Perception is emulated as occurring in stages or "layers" that proceed from input (sensation) through output (response). For instance, the processing of auditory messages is a three-stage process:

1. Hearing, which is initiated by auditory input
2. Message-understanding, which is a cognitive process initiated by hearing
3. Attending, which represents the hearer's attending and reacting to the spoken message.

The particular demands of aircrews suggest a fourth auditory process: ignoring messages that are broadcast on the ATC network but that are directed toward other aircraft. Similarly, visual processing is a two-stage process. The first stage is *identifying* the visual event, and the second is *responding* to the event by triggering the set of appropriate procedures. An additional *reading* stage is used to simulate the ATC operator's activities required to understand the situation when he/she takes over control of airspace from the

previous controller at the start of a shift. This stage processing interpretation implies that motor activity is a part of perception. In essence, motor actions close the loop on the perceptual categorization process.

Much like OMAR's lack of an executive processor, no single process or component represents attention (Deutsch, 1997). Rather, attention is modeled as an emergent quality from the proactive and reactive reactions of sensory processing.

The schema-like constructs of Simple Frame Language (SFL), a knowledge representation system, emulate the organization and function of LTM. However, the model does not posit a global short-term store. Instead, it assumes that performers are experts who access LTM directly through recognition processes. For instance, the model simulates the memory required to monitor aircraft (e.g., remembering the position of an aircraft's icon on a radar scheme) by having the performer constantly revisit each aircraft.

OMAR uses Flavors Expert (FLEX), an expert system toolkit, to model decision-making as a rule-following process. However, it does not emulate more complex problem-solving processes nor does it simulate the effects of learning.

The multiagent architecture of OMAR is especially suited to model social interactions. For the ATC model, OMAR models in-person interactions among aircrew members, "party-line" radio communications among aircrews, and telephone communication between ATC centers in adjacent sectors.

4. Applications

OMAR has been used primarily to simulate behavior related to ATC. Within that environment, OMAR agents have modeled a variety of players, including different members of aircrews as well as controllers.

Outside of ATC applications, D-OMAR has been used to provide synthetic team members for the Distributed Dynamic Decision-making (DDD) simulator under development by Aptima, Inc. (2001). In this application, DDD was used to simulate the interactive environment of Airborne Warning and Control System (AWACS). D-OMAR has also been employed as the simulation backbone of the Air Operations Enterprise Model (AOEM), which emulates command and control (C2) organizations and processes involved in planning and preparation of air operations (Brown, 2000).

As described earlier, D-OMAR has also been used to provide the simulation environment for the AMBR project (Deutsch and Benyo, 2001). In that particular project,

D-OMAR played the part of the simulation, not a candidate cognitive model. The simplified ATC task used in the AMBR project was also used to evaluate display characteristics for decision support systems (MacMillan, Deutsch, and Young, 1997).

5. Technical Considerations

a. Input and Input Aids

According to the User/Programmer Manual (Freeman, 2002), the user's primary input is the specification of the simulation scenario. With respect to ATC applications, three aspects of the simulation scenario must be specified:

1. **Agents and their tasks.** The user must specify all players and the behaviors that they perform in the scenario, including a specification of goals and rules that govern those behaviors.
2. **Effects of communications on agents and tasks.** Users must specify how and when communications interrupt ongoing activities.
3. **Relevance of communications.** Users must specify, for each agent, the types of communications that are directly relevant to enable them to focus their attention.

To aid in scenario and model development, OMAR includes a Developer's Interface, which includes a suite of software tools used in developing human performance models or agent-based systems. For data input purposes, this graphics-based toolset includes a Concept Editor for defining SFL objects, and a Procedure Browser for displaying a graphic view of the structure of goals and procedures.

b. Model Output and Analysis Tools

The Developer's Interface also includes output and performance analysis tools. The first is a Simulator Control Window from which the user can monitor the activities and events in the simulation as they occur. After the simulation occurs, the OMAR Time Line summarizes simulation events either as a task time line, which is a GANTT-style chart that provides data on simulation goals and procedures, or an event time line, which organizes event data by agent. Finally, a Post-Run Analysis Tool permits the user to perform analyses of scenarios that were run previously.

c. Computer Language

According to Deutsch (2002), OMAR uses three different knowledge representation languages:

1. **SFL (Simple Frame Language).** SFL is a direct descendent of KL-ONE and is used to define all objects in the simulation world. In OMAR, SFL provides the function of declarative knowledge.
2. **Simulation Core (SCORE).** Based on ACTORS semantics (Agha, 1986), SCORE is a procedural language for defining agents used to represent individuals or capabilities within individuals. SCORE tasks are SFL objects.
3. **FLEX (Flavors Expert).** FLEX is an expert system toolkit used in OMAR to simulate rule-based behavior and decision-making.

The original OMAR software has evolved into two forms. The first is referred to as OmarL, or the LISP implementation of OMAR. This is an update of the original event-based OMAR simulation, which was written in ACL (Allegro Common LISP) with the graphics in Common LISP Interface Manager (CLIM). The present version of OMAR (4.0) requires two additional software components: ACL Version 6.1, which is a product of Franz, Inc., and the Java Software Development Kit (SDK) Version 1.3.1 or later, which can be downloaded from the Sun Web site (http://java.sun.com) for free.

The other form is OmarJ, which is the new Java implementation of D-OMAR. (D-OMAR was originally written in LISP.) OmarJ is not an HBR model per se; rather, it is a system for managing systems of agents. OmarJ requires the Java SDK 1.3 or better to be mounted on the machine that runs OmarJ. From Web site http://omar.bbn.com, OmarL and OmarJ can be downloaded, and relevant documentation can be obtained.

d. Interfaces

To facilitate communication with other applications, D-OMAR is constructed in "layers" (Cramer, 1998). At the core of the system is the original version of OMAR, written in LISP (OmarL). To this core, D-OMAR adds OmarJ, which provides the following key components:

- A procedural language (ScoreJ) for controlling parallel program execution
- A time management system (scorej.OmarClock) that keeps track of executing threads
- An inter-process communication mechanism (Signals) for coordinating thread execution

- An automatic event recording capability (not available in current version)
- An external communications system.

The latter component, an external communications system, is the key component of D-OMAR for interfacing with other models and simulations. The user has a choice of three implementations:

1. A Java implementation of the standard D-OMAR protocol, where objects are sent across a "socket" by first sending a data type code, followed by the data.
2. A Jini network implementation that allows OmarJ nodes to use the Jini network to send signals back and forth and to interface with non-Omar Jini nodes.
3. A Control of Agent-Based Systems (CoABS)[16] grid built on top of a Jini network.

The capability of D-OMAR to be interfaced with other HBR models was demonstrated in the AMBR project, where it was used to provide the ATC simulation environment. In that project, D-OMAR was successfully linked with the Air Force's DCOG, ACT-R, COGNET/iGEN, and EPIC-Soar.[17] D-OMAR has also been used to control Jack®, an animated, interactive virtual simulation of a human (Deutsch, MacMillan, et al., 1997).

6. Evaluation

Of all models, OMAR offers the most elaborate and validated system for interfacing with other models and simulations. OMAR is also one of the few models that simulates the interactions among multiple performers. On the other hand, OMAR (or at least, OmarL) has not been tested as a standalone HBR model outside the context of ATC (air traffic control).

---

16 CoABS is a Defense Advanced Projects Agency (DARPA)-sponsored system that provides many services useful to agent systems.

17 EPIC-Soar is an integrated hybrid model of EPIC and Soar.

Q. PSI

1. Model Purpose and History of Development

PSI[18] is a computational theory of psychology that focuses on the interaction of cognitive, emotional, and motivational processes. It is currently under development by Dietrich Dörner and his colleagues at the Institut für Theoretische Psychologie der Otto-Friedrich-Universität Bamberg. According to Bartl and Dörner (1998), the purpose of the PSI project is to create "…an intelligent, motivated, emotional agent…which is able to survive in arbitrary domains of reality." The developers have constructed domains specifically to compare the behavior of PSI agents with the behavior of actual humans.

The University of Bamberg (Germany) currently maintains a Web site on PSI at http://www.uni-bamberg.de/~ba2dp1/psi.html. PSI software is available at this site, but all the documentation and most of the additional information at the site are provided in German. The following review is based almost entirely on the few English papers that have been published on their Web site (Dörner and Hille, 1995; Bartl and Dörner, 1998; Strohschneider, 2002) and on the Ritter, Shadbolt, et al. (2002) review of PSI and other cognitive models developed in Europe.

2. Principal Metaphors and Assumptions

The central psychological construct in PSI is motivation, which is represented by a homeostatic or hydraulic metaphor. Motivators are portrayed as tanks filled with liquids, which must be kept within certain tolerance levels. When the level deviates from the ideal, a motivator is launched to activate behaviors to restore the levels. Bartl and Dörner (1998) identify six different motivators, which are sorted into three categories of needs as follows:

I. System needs intended to address an organism's existential requirements:
   (1) Water
   (2) Energy

---

[18] PSI is usually presented in all capital letters, but has not been defined as an acronym. According to Dietrich Dörner, "PSI is not an acronym. It just is the first letter of the Greek word for "soul". And this is because it is our intention with the PSI-project not only to simulate cognition, but motivation, emotion and what we call "action regulation" too. Just the whole soul! And that's the idea behind PSI" (personal communication, December 29, 2003).

II. Preservation needs designed to maintain an organism's structures:

(3) Pain avoidance

III. Information needs that have a more cognitive or social basis:

(4) Certainty

(5) Competence

(6) Affiliation.

The multiple needs interact, sometimes in complex fashion, to activate specific behaviors. For example, consider the interaction of two information needs (certainty and competence). A need for certainty is increased when PSI learns that the results of an action have not been predicted correctly, whereas a need for competence is satisfied when other needs (e.g., water) are fulfilled. Low certainty levels may launch exploratory behavior, but, if certainty and competence are low, PSI may elicit flight. On the other hand, moderately low competence may elicit "adventure-seeking" behavior to prove one's competence, but excessively low competence may not.

Action strategies are governed by Rassmussen's (1983) three-level hierarchy of information processing. That is, PSI first tries to satisfy system goals using automatized skills. If this is not successful, PSI resorts to knowledge-based behavior, including schemata-like structures. Finally, if all else fails, the system uses trial-and-error to accomplish goals.

Several needs can be active at once. A motive selection mechanism designates a single need as the actual intention—that is, the activated motive. The mechanism simply selects the intention with the highest expectancy value, which is defined as the product of the perceived probability of fulfilling the need and the level of need. The resulting product is referred to as motive "strength."

In PSI, emotions and cognitions are not separate processes; rather, they are the result of external situations and act to modulate both processes. The primary mechanisms for shaping or modulating cognition and motivation are as follows:

- **Activation level.** The strengths of various needs lead to specific behaviors and to an increase in general activation level, which speeds information processing and may trigger either or both of the two modulators described below.
- **Resolution level (RL).** Perceptions are modeled as comparisons among schemata. RL refers to the required exactness of those comparisons. At low,

general activation levels, RL is high, which results in slow but reliable processing. At high activation levels, RL is low, which leads to fast but inaccurate processing.

- **Selection threshold (SL).** SL refers to the ability of PSI to change dynamically the threshold needed for activating a need. This mechanism effectively defends intentions against competing needs, thereby protecting PSI against strong behavioral oscillations.

To illustrate the role of emotion in PSI, Dörner and Hille (1995) use "anger" as an example. The external event correlated with the emotion "anger" is the unexpected hindering of a system goal (Bartl and Dörner, 1998). This experience triggers the following prototypical styles of information processing:

- High activation level (i.e., the person processes information at a high rate of speed)
- Low RL (decisions are less likely to take current conditions into account).
- High SL (behavior is resistant to change)

In addition, the person is likely to update the image of the actual situation at a low rate, so that they are relatively unresponsive to changes in the environment. Under these conditions, "…it is appropriate to be angry" (Dörner and Hille, 1995, ¶ 13).

3. Cognitive/Behavioral Functions Represented

According to Ritter, Shadbolt, et al. (2002), visual perception is modeled through a "Hypercept" process that

> …scans the (simulated) environment for basic features. It raises hypotheses about the sensory schemas to which the features may belong and tests these hypotheses by subsequent scanning of the environment (comparable to saccadic eye-movements). If a pattern is not recognizable, a new schema is generated (p. 45).

With regard to perceptualmotor functions, Ritter, Shadbolt, et al. (2002) indicate that basic actions can be combined into complex sensory-motor programs through learning.

Memory is a simple log of perceptions and activities, and forgetting is modeled as a decay of that record. Forgetting is important because it acts to make schemata increasingly abstract over time. STM is modeled as the "head" of the record. Elements in STM are transitioned in continuous fashion to an episodic memory, and the remnants of the record (stripped of detail) are eventually transitioned into LTM. Emotions interact with

memory in that records associated with need satisfaction or with pain have a greater chance of surviving to LTM than do simple sequences of events.

According to Ritter, Shadbolt, et al. (2002), PSI can model a wide variety of learning situations, including associative and perceptual learning, operant conditioning, sensory-motor learning, goal learning (i.e., remembering situations that lead to need satisfaction), and aversions (i.e., remembering situations or needs that cause needs).

Decision-making processes are modeled as expected utility problems. In addition, according to Ritter, Shadbolt, et al. (2002), PSI includes several built-in problem-solving strategies, including hill climbing and trial and error.

4. Applications

Bartl and Dörner (1998) modeled PSI's play on BioLab, a computer-based game that depicts the biological production of sugar beets for energy production. Comparisons between PSI and humans (college students at the University of Bamberg) indicated that PSI and humans were very similar in performance on component processes of the game. However, the best humans were able to beat the PSI model in overall production. The authors speculated that the difference was caused by the fact that PSI had no language or metacognitive capability to contemplate its own actions and strategies. They also contended that such capabilities could easily be modeled in PSI.

5. Technical Considerations

a. Input and Input Aids

The generic tools associated with Delphi 4, the implementation language for PSI, can be used to develop PSI models. However, the exact input requirements for PSI are not clear because die Unterlagen werden auf Deutsch geschrieben (the documents are written in German).

b. Model Output and Analysis Tools

Output from PSI includes momentary states of motives and the speed and accuracy of simulated behavior. The current model of PSI is personified by a cartoonish tiny blue steam engine that chugs across a landscape in search of food, water, and nuggets of gold. This model is able to learn from experience and express emotion through an animated face that is projected along with the PSI "creature."

Currently, no known output analysis tools are associated with PSI.

c. Computer Language and Interfaces

According to Ritter, Shadbolt, et al. (2002), PSI is implemented in Delphi Pascal in a Microsoft Windows environment. The source code can be downloaded from the following Web site: http://www.uni-bamberg.de/ppp/insttheopsy/psi-software.html.

No known examples exist of interfacing PSI with other models or simulations.

6. Evaluation

PSI is a complex system that is difficult to evaluate for two reasons: Most of the literature concerning the model is published in German, and it is far from a complete model of human cognition (Ritter, Shadbolt, et al., 2002). On the other hand, it is one of the few HBR models that even attempts to integrate motivation and emotion with cognitive processes.

R. SITUATION AWARENESS MODEL FOR PILOT-IN-THE-LOOP EVALUATION (SAMPLE)

1. Model Purpose and History of Development

The original purpose of the Situation Awareness Model for Pilot-in-the-Loop Evaluation (SAMPLE) was to provide a model of individual and crew SA in air combat. The purpose of SAMPLE has expanded to include modeling SA in other types of performance environments.

SAMPLE was developed by Greg L. Zacharias, Karen A. Harper, and their colleagues at Charles River Analytics (CRA), Inc. According to Mulgund et al. (2000), SAMPLE has three different modeling predecessors:

- The first predecessor to SAMPLE was the Optimal Control Model (OCM) (Kleinman, Baron, and Levinson, 1971), which is an information-processing model for simulating performance on continuous, closed-loop tasks. The sensory limitations and information-processing models from OCM were adopted in SAMPLE.
- The second predecessor to SAMPLE was the Procedure-Oriented Crew Model (PROCRU), which integrated continuous movement with discrete procedural actions of crews (Baron et al., 1980). Particularly relevant to the development of SAMPLE, Zacharias, Baron, and Muralidharan (1981)

developed a PROCRU model of crew performance on anti-aircraft artillery tasks that made the task of situation assessment an explicit part of the simulation.

- The third predecessor to SAMPLE was the Crew/System Integration Model (CSIM), which provided structural formalism for many of the concepts in the two previous models (Zacharias and Baron, 1982). SAMPLE incorporated many aspects of the information processing, situation assessment, and decision-making modules of CSIM and integrated with Rasmussen's (1983) three-tier model of information processing and skilled behavior.

CRA continues to develop SAMPLE and related products under the sponsorship of the AFRL's Human Effectiveness Directorate. An interim delivery of the enhanced SAMPLE architecture and tools was scheduled for July 2002, and final delivery was scheduled for June 2003 (K. Harper, personal communication, May 28, 2002).

2. Principal Metaphors and Assumptions

One of the more fundamental assumptions of SAMPLE is that situation assessments and the decisions based on those assessments are driven by a pattern recognition process as described by Klein's (1989) Recognition-Primed Decision-making (RPD) model. In RPD, decisions are based on rapid assessments of the current situation. This model is viewed in stark contrast to classic decision-making theories in which the performer deliberately assesses utilities associated with alternative courses of action.

The SAMPLE model also assumes, in accord with Endsley's (1988) concepts, that SA can be decomposed into three increasing levels of awareness:

- **Level 1 (detection).** Detection is the lowest level of SA and is defined as the perception of key elements of the situation in space and in time. In other words, it refers to perception of the elements of the current situation.
- **Level 2 (identification).** The middle level of SA is the integration of the elements perceived in Level 1 into an understanding of the situation.
- **Level 3 (prediction).** Prediction represents the highest level of SA, which is the projection of the current situation into the near future.

Like other HBRs, SAMPLE incorporates a stage model of information processing. The details of information flow are consistent with Rasmussen's (1983) three-tier model of processing:

- **Skill-based behavior.** This is the least complex level of processing and is viewed as a data-driven pattern-recognition process. This behavior pertains to well-practiced skills and involves little or no higher order processing.
- **Rule-based behavior.** If the pattern recognition process fails to identify an associated response, the model searches a set of rules, which are viewed as compiled situation-action pairs. This behavior is characterized by the use of standardized procedures.
- **Knowledge-based behavior.** This applies when the performer faces new or unusual situations for which no standardized procedures exist. In this case, the performer invokes problem-solving processes to address the task.

The architecture of SAMPLE is divided into three stages of processing: information processing, situation assessment, and decision-making/procedure execution (Zacharias et. al, 1996; Mulgund et al., 2000).

- **Information Processing.** The information-processing stage comprises two submodels—a continuous state estimator and a discrete event detector. In accord with OCM (Kleinman and Baron, 1971), the state estimator predicts continuous data via a Kalman filter. The output from this submodel bypasses the next two stages and feeds directly into a task execution component. This architectural feature of SAMPLE is intended to model skill-based behavior.

  The discrete event detector, which is simulated by a fuzzy/Boolean logic system, transforms sensory data into situationally relevant semantic variables or cues. This aspect of SAMPLE simulates the use of contextual knowledge that is typical of human task performance. In contrast to the state model, output from the event model is sequentially processed by the next two processes (situation assessment and decision-making). This aspect of the model is intended to emulate rule-based behavior.
- **Situation Assessment.** This stage emulates information fusion and reasoning required in a multitask environment. The agents for this process are based on Bayesian belief networks, which represent knowledge using nodes and links that carry and modify information propagated among nodes. This network constitutes a high-level representation of and memory for perceived events. The assessment process involves matching events passed on from the information processor with these stored representations. All three of Endsley's levels of SA are incorporated in this process. These representations include pre-programmed *a priori* knowledge and updates to SA that propagate through the network in simulated real time.
- **Decision-Making/Procedure Execution.** The final stage is represented by two submodels: a procedure selector and a procedure executor, which

represent rule-based decision processing and psychomotor skill performance, respectively. The decision-making component is simulated by an expert system that employs production system rule sets. The model assumes that the system's rule bases are sufficient to address all situations; thus the third level of Rassmussen's hierarchy (knowledge-based behavior) is not explicitly modeled by SAMPLE.

Psychomotor performance is represented by the conversion of selected procedures into information required to control the vehicle. These motor-oriented data are organized into three channels: information, controls, and communications.

3. Cognitive/Behavioral Functions Represented

Perceptual effects are modeled as the translation of sensory data (i.e., state or event information) into symbolic information relevant to the situation. Delays in onset of motor actions are modeled, but other motor-related details are not.

Attention is explicitly modeled as a subcomponent of the information-processing stage. The attention subcomponent is a selection process that reflects the inability of the human to process all available information. The selective attention process is under the control of the decision-making stage, which ensures that attention is focused on information directly relevant to current tasks and procedures. Harper and Zacharias (2002) point out two limitations of SAMPLE's attention model:

1. The model allows the simulated operator to perform multiple tasks without suffering a concomitant degradation in performance.
2. Although attention adjusts the field of view in accordance with task goals, the agent processes all information within the view to the same level of detail and accuracy.

SAMPLE includes four separate memory stores that serve as repositories for perceptual events, situational assessments, decision rules, and procedures. The model does not distinguish between an STM/WM and a more permanent LTM. Further, the model assumes that the performer knows all procedures (i.e., is an expert) but adapts behavior on the basis of the current situation. Thus, no learning effects are modeled.

Decision-making is modeled as a reactive, rule-based process. Problem solving, a knowledge-based behavior in Rasmussen's parlance, is not modeled in SAMPLE.

The agent architecture of SAMPLE enables the user to emulate multiple performers. In Mulgund et al. (2000), developers modeled a four-aircraft situation (two red, two

blue), where the blue pair of aircraft pilots share their situational awareness to allow the leader to direct other flight members in engaging the threat.

4. Applications

SAMPLE was originally designed to model situation assessment for the tactical aviation pilot. However, the developers insist that SAMPLE is domain independent, and it has since been applied to several different contexts, including

- Nuclear power plant operations (Zacharias et al., 1994)
- Commercial aviation (Harper et al., 1998)
- Military operations in urban terrains (Harper et al., 2000)
- Control of unmanned aerial vehicles (UAVs) (Hanson, Sullivan, and Harper, 2001)
- Decision-making of individual infantrymen and of fireteam and squad leaders (Aykroyd et. al, 2002).

5. Technical Considerations

a. Input and Input Aids

The fundamental inputs to a SAMPLE model are the results from a cognitive task analysis (CTA).[19] To incorporate the input from CTAs, the results are organized as a feed-forward network of nodes and links that describe the mental model of the expert performer. Three types of nodes in that network correspond to theoretical mechanisms in SAMPLE: fuzzy system, belief network, and expert systems. In addition, SAMPLE supports I/O modules that act as interfaces for data going into or coming out of the model. Generic nodes allow the user to include object-oriented code for functions that are not easily modeled by fuzzy systems, belief networks, or expert systems.

To aid users in developing SAMPLE models, CRA has developed a software utility called the Graphical Agent Development Environment (GRADE). One of the GRADE tools is a graphic aid for developing the overall organization (the topology) of the mental model. Separate tools exist or are proposed for developing the fuzzy system belief network and expert system components of the model (Harper et al., 2001).

---

[19] Of the numerous extant CTA methods, Hanson et al. (2002) suggest that the one that is most compatible with SAMPLE models is the so-called Goal-Directed Task Analysis (GDTA) method, which was specifically designed by Endsley and associates to identify the SA requirements of tasks.

b. Model Output and Analysis Tools

SAMPLE produces records of activities and time lines for all three model components (information processing, situation assessment, and decision-making). One unique source of output is based on the distinction between two pilot models used by SAMPLE: a *reference* pilot who receives perfect information about the situation state and events but does not control the aircraft and an *acting* pilot who receives information through onboard systems (radar, displays, and so forth) and controls the aircraft accordingly. SAMPLE compares data produced by both models to produce three types of metrics, corresponding roughly to the three SA levels as defined by Endsley (1988):

- **Information disparity.** This measure focuses on pilot errors in aircraft state estimation.
- **SA disparity.** This measure determines the difference between actual and assessed belief values.
- **Combat advantage index.** This is an index of the advantage that the ownship has over its target in altitude, speed, and geometric potentials. It is normalized to vary from +1 (greatest ownship advantage) to –1 (greatest ownship disadvantage).

GRADE will also include the capability to step through the model and freeze the agent state to examine current perception and knowledge states (Harper et al., 2001). This feature will presumably aid the developer in validating the model with subject matter experts (SMEs). GRADE also will also provide a feature to visualize and trace model performance after execution. This feature affords another level of model validation and analysis.

c. Computer Language and Interfaces

Interagent communication is accomplished through the Command and Control Simulation Interface Language (CCSIL), a special language developed by the DARPA Command Forces (CFOR) project to model communication among entities in the Distributed Interactive Simulation (DIS) environment. According to Harper et al. (2001), future plans call for a more generic approach to agent communication, such as eXtensible Markup Language (XML).

According to Mulgund et al. (2000), the basic SAMPLE software was implemented using object-oriented C++ on a UNIX workstation. The other components are implemented as follows:

- Fuzzy logic reasoning is performed using software derived from the Matlab Fuzzy Logic Toolbox.
- Belief network modeling is implemented using Bnet, a stand-alone C++ system developed by CRA, Inc.
- Expert system functions are performed using the C Language Integrated Production System (CLIPS) expert system shell.

SAMPLE has been used to provide the HBR model in at least two military simulations:

1. Boeing's Man-in-the-Loop Air-to-Air System Performance Evaluation Model (MIL-AASPEM) (Mulgund et al., 2000; Hanson et al., 2002)
2. Simulation Technology, Inc.'s Integrated Unit Simulation System (IUSS) (Aykroyd et al., 2000).

SAMPLE has also been used to control the behavior of virtual humans by interfacing with Boston Dynamic's DI-Guy™. SAMPLE has not been interfaced with other cognitive models, however.

6. Evaluation

SAMPLE is good at modeling the "front-end" of cognition—particularly SA. It does not, however, model some of the higher-level processes, such as learning and problem solving. At the output end of cognition, SAMPLE provides a relatively undetailed model of psychomotor performance.

S. STATE, OPERATOR, AND RESULT (Soar)

1. Model Purpose and History of Development

Soar, which formally stands for State, Operator, And Result,[20] is perhaps the most popular HBR model gauged by the number of adherents. As its name implies, Soar's concept of cognition involves a search through a problem space and application of operators to states in order to achieve a result. Part of Soar's popularity can be traced to the fact that it is a multifaceted model that addresses disparate audiences. Ritter, Baxter, et al. (2002) articulate three different, yet interrelated, purposes for Soar:

---

[20] According to Ritter, Baxter, et al. (2002), the Soar developer community stopped regarding Soar as an acronym. Hence, it is not usually written in all caps.

1. Soar represents a unified theory of cognition. This vision of Soar emphasizes its heuristic value for developing theories of cognition.
2. Soar embodies a set of principles and constraints derived from its theory of cognition. In this sense, Soar provides an integrative cognitive architecture from which one can construct applied models of knowledge-based behavior, including problem solving, learning, and human interaction with external systems and environments.
3. Soar is an AI language. This view implies that Soar is a technology that supports the development of HBR models.

Soar has its roots in work begun in the 1950s by Allen Newell, J. C. Shaw, and Herbert Simon to demonstrate that computers could address complex problem solving. The first model produced by this group was the Logic Theorist (LT), which was designed to devise proofs of geometry theorems (Newell and Simon, 1956; Newell, Shaw, and Simon, 1957). Those same researchers extended the ideas of the LT to different types of problems in the GPS (General Problem Solver) model (Newell, Shaw, and Simon, 1958; Newell and Simon, 1972).

Soar was born from the Newell's specific desire to allow users to write directly onto problem spaces (Carley and Wendt, 1991). In 1982, Newell's graduate student, John Laird, tried to avoid the complications of production systems and implemented the problem space in a LISP-based system called Task Experimenter (TEX), which subsequently became known as Soar 1. Because TEX lacked the control structures for adding knowledge, the next version (Soar 2), developed in 1983, was implemented using production systems software (Laird and Rosenbloom, 1994). The principal theoretical improvement of Soar 2 was its ability to acquire knowledge through a process known as subgoaling. Another graduate student, Paul Rosenbloom, joined the team in 1983, and he and Newell successfully applied the Soar model to problems in computing and cognition.

The first public paper on Soar was presented by Laird and Newell in 1983. Work on the system spread quickly to other sites as the graduate students moved on to other institutions. Rosenbloom (1994) provided this summary of the first decade of work on Soar in the United States:

> Soar started as a standard collaboration between a faculty member (Newell) and a graduate student (Laird). In '83, John and I finished our degrees and agreed to stay on at CMU and work with Allen on pushing Soar as a general cognitive architecture. In '84, John and I both left CMU and moved to the San Francisco Bay Area [Rosenbloom to Stanford University

> and Laird to the Xerox Palo Alto Research Center (PARC)]. We kept up the collaboration on Soar, but now at a distance. The first workshop occurred in '86 (at Stanford) and was instigated by the fact that we now had two semi-independent groups working on Soar (one at CMU and one at Stanford/Xerox), and we needed to increase the interactions among them to maintain a coherent project. We've continued to hold workshops in the US approximately every 8–10 months since then. The first few alternated between Stanford and CMU. Michigan was then added to the mix after John moved there in '86. USC/ISI [University of Southern California/Information Science Institute] replaced Stanford after I moved here in '87. Ohio St. later joined the rotation after a critical mass of Soar researchers grew up there (and they showed interest in holding workshops) (¶ 1).

As Soar propagated throughout the United States, Soar communities were also spreading to Europe. Rosenbloom (1994) associates these developments with visits by several European researchers to CMU—in particular, John Michon of the University of Groningen in the Netherlands and Richard Young of the Medical Research Council in Cambridge, England. The first European Soar Workshop was held in 1988 in England, and subsequent events have been held at various sites on the continent. Soar also has three "official" academic sites in England at the Universities of Nottingham, Hertfordshire, and Portsmouth.

From 1992 to 1997, DARPA sponsored further development of Soar in a project involving the groups at CMU, at USC/ISI, and at the University of Michigan Artificial Intelligence Laboratory. The purpose of the project, which was called Soar/Intelligent Forces (IFOR), was to develop intelligent automated agents for tactical air simulation. Two models were developed for Soar/IFOR:

1. TacAir-Soar, which models a variety of fixed wing air-to-air and air-to-ground missions
2. RWA-Soar, which models rotary-wing aircraft missions.

The first operational test of TacAir-Soar was in the context of the Synthetic Theater of War-Europe (STOW-E) training exercise. This was a theater-level exercise held at 18 sites. TacAir-Soar "flew" friendly blue sorties and provided opposing force aircraft to pit against friendly pilots in virtual simulations. The final test was STOW-97, during which TacAir-Soar and RWA-Soar provided all air missions as specified in the air tasking order.

At the conclusion of the Soar/IFOR project, some team members formed Soar Technology, Inc. to continue the development of intelligent synthetic forces for defense applications and to explore applications in other areas, such as commerce, finance, security, and consumer goods. Founded by John Laird, Soar Technology, Inc. is located in Ann Arbor, Michigan, and employs a diverse staff of computer scientists and cognitive psychologists to achieve those goals.

Soar is also a core technology employed by the Institute for Creative Technologies, which was founded by Paul Rosenbloom and others. Affiliated with the University of Southern California, this research center was started in 1999 with a 5-year, $44.3 million contract with the U. S. Army's Simulation, Training, and Instrumentation Command (STRICOM).[21] The purpose of the center is to incorporate technologies from the entertainment and game industries to create immersive and compelling simulation environments for training applications.

The first purely commercial application of Soar was the rule engine called Knowledge-Based Agent (KB Agent). KB Agent is a product of ExpLore Reasoning Systems, Inc., which was formed in 1995 to help organizations build automated decision aids. KB Agent uses the Soar architecture to provide a tool for modeling and implementing business expertise that is based on a model of human cognition. The purpose is to go beyond traditional rule-based systems by providing facilities for hierarchical abstraction, search, and learning.

Soar is currently under active development at several sites around the world. More information about academic efforts in the United States can be obtained at the following Web sites:

- The University of Michigan:
  http://ai.eecs.umich.edu/soar
- Carnegie Mellon University:
  http://www-2.cs.cmu.edu/afs/cs/project/soar/public/www/home-page.html
- The University of Southern California:
  http://www.isi.edu/soar/soar-homepage.html.

Three other Soar Web sites are associated with universities in England:

1. The University of Nottingham:

---

[21] STRICOM has since been renamed the Program Executive Office for Simulation, Training, and Instrumentation (PEO STRI).

http://www.nottingham.ac.uk/pub/soar/nottingham/soar-faq.html

2. The University of Hertfordshire:
   http://phoenix.herts.ac.uk/~rmy/cogarch.seminar/soar.html
3. The University of Portsmouth:
   http://www.dcs.port.ac.uk/~hirsta/soarteam.htm.

Finally, information on Soar development is also available from two commercial Web sites:

1. Soar Technology, Incorporated:
   http://www.soartech.com
2. Explore Reasoning Systems, Incorporated:
   http://www.ers.com.

2. Principal Metaphors and Assumptions

The development of Soar is explicitly constrained by three general assumptions about human cognition and behavior: Behavior is flexible and goal-driven, learning occurs continuously from experience, and elementary cognitive processes occur well within one second (Lewis, 2001).

Another guiding principle of Soar is that it should comprise a small set of orthogonal mechanisms (Rosenbloom et al., 1991). This assumption drives the model not only toward simplicity, but also toward uniformity in architecture. For instance, the Soar has a single type of structure or process for LTM structure, learning, task representation, and decision-making.

Harking back to some of the original work on GPS and other AI approaches to problem solving, Soar depicts all behavior as movement through problem spaces. A problem space defines the states and operators that apply to the task at hand. The knowledge required to execute tasks are modeled as production rules, which are condition-action pairs. The conditions' component of productions define access paths to knowledge stored in memory, whereas the action component defines the memory contents themselves (Lewis, 2001).

The course of information processing in Soar is described by the "decision cycle." Hill (1999) describes the decision cycle as a four-phase iterative process:[22]

1. **Input.** This phase mimics human perception. Input productions take information from the external world and place the contents into WM.
2. **Elaboration.** Productions in LTM are matched against the contents of WM and fire in parallel so that all relevant knowledge is retrieved. The process continues until no more rules match and productions cease firing (quiescence). In this phase, the contents of WM are not changed. Rather, these productions have two outcomes: They create "preferences" or "proposals" for actions that are evaluated in the decision phase, and they issue direct commands to the motor system.
3. **Output.** Motor commands are executed. Resulting changes in internal and external conditions are considered during the decision phase.
4. **Decision.** Proposals for action are examined and, as a result, the system selects appropriate operators. If no such action is called for or several competing actions are indicated, Soar recognizes an impasse, which automatically sets up a subgoal (creates a new space) for resolving the impasse. If the subgoal recognizes another conflict, another subgoal is declared for solving that impasse, creating a goal stack. This process proceeds in iterative fashion until all impasses have been resolved. Once the decision phase ends, the cycle begins again.

3. Cognitive/Behavioral Functions Represented

In Soar, perception is represented by encoding productions that take data off of a perceptual buffer (called the input link) and place the results into WM. Similarly, motor actions are represented by decoding productions that translate WM elements into a form that the motor system can use and place results on an output link. Perceptions and actions are modeled at the entity level—that is, the virtual performer "perceives" or "acts" with direct access to a fixed set of object attributes, such as entity type, location, color, orientation, and so forth. Sensory models are used to filter what information is potentially perceptible. For instance, entities that are beyond visual range are not perceptible by Soar.

---

[22] The Soar decision cycle has variously been described as a 2-cycle (elaborate-decide) process (Laird and Rosenbloom, 1994; Lehman, Laird, and Rosenbloom, 1998), a 3-cycle (recognize-decide-act) process (Lewis, 2000), and even a 5-phase (input-proposal-decision-application-output) process (Laird et al., 1999). Hill's 4-stage description appears to provide sufficient details while capturing the iterative nature of the process.

Also, in a manner very much like ACT-P/M (described in Section II.A), Chong and Laird (1997) supplemented Soar with EPIC to enhance its perceptualmotor models.

The default version of Soar does not constrain the number of percepts that can be put into WM. For aviation applications of Soar, this has had the consequence of overloading the virtual pilot with perceptual processing requirements and periodically losing control of his aircraft (Hill, 1999). Thus, to the extent that the attention mechanism limits information, it has had a practical and a theoretical value to the model. In this aviation version of Soar, attention provides two functions: focusing perception on a subset of perceptible objects by instantiating a zoom lens metaphor and grouping objects so that perception is based on collections as opposed to individual entities. Attention is controlled by a top-down process, where focusing and grouping are driven by task goals, and by a bottom-up process, where attention is captured by external stimuli.

SA for the same aviation application is viewed as an elaboration on the attention mechanism (Zhang and Hill, 2000a; 2000b). The primary mechanisms for SA are templates, which are hierarchical representations that can be used to match against the current situation. The templates are used to control attentional focus (i.e., the extent of zoom) required by the current situation. The templates are also used to control the pilots' search. The outcomes of the search processes are used to confirm or deny hypotheses that the virtual pilot has about the situation.

All LTM (procedural, declarative, episodic) is stored as production rules. (Approximately 1,400 rules are used in TacAir-Soar.) Productions are used not so much to model rule-based behavior, but to provide a form of content addressable memory, with the conditions' component of productions providing associative pathways to contents contained in the production action. LTM is accessed in parallel such that all relevant information is retrieved before Soar's decision cycle is completed. Productions can be added to LTM through learning (i.e., chunking), but no procedures exist for deleting productions from LTM.

WM provides a store of elements that represent the current situation. This mechanism provides the nexus of information processing in Soar because it integrates inputs from the external world, from information in the actions of productions (i.e., the contents of LTM), and from results of Soar's internal decision processes. WM is short-term in the sense that it contains information related to the present situation, but it is not limited in capacity or time. The WM elements are a set of interrelated objects, where objects are

attribute-value pairs. Objects are interrelated by sharing attributes and values with other objects.

Soar posits a third type of memory, preference memory, which stores suggestions or imperatives about current operators (Laird, Congdon, and Coulter, 1999). Preferences are encoded according to a fixed semantics. This mechanism was developed to support the decision stage in Soar information processing. Rosenbloom et al. (1991) view preference memory as a special type or subset of WM.

Decision-making is explicitly modeled in Soar. The model represents deliberative and reactive decision-making. Deliberative processes are required when the virtual performer does not have sufficient knowledge to execute a task and must form new subgoals to attain the requisite knowledge. Reactive decisions, in contrast, are those that do not require the formation of new subgoals and are based exclusively on knowledge of the current situation.

Soar models two types of planning methods corresponding to weak and strong approaches to problem solving. The weak planning method (i.e., generalizable to a range of situations) is the look-ahead search process that Soar employs when forming subgoals because of an impasse. The strong planning method, in contrast, is specific to military planning techniques. For example, Hill et al., (1998) described a planning routine for the virtual commander of attack helicopter battalions in the STOW-97 Advanced Concept Technology Demonstration (ACTD).[23] This particular planning agent organized company plans according to a hierarchical graph structure and performed three key activities:

1. **Plan generation.** The virtual company commander takes a battalion operations order (composed by a human role player), decomposes it into primitive tasks, and generates additional tasks designed to satisfy the preconditions stated in the battalion order.
2. **Plan execution.** Using the completed plan and the current world description as input, execution operators control the initiation and termination of company tasks.
3. **Replanning.** The planning agent monitors plan execution and triggers replanning when it recognizes an unexpected event. The plan can be repaired by either extending the existing plan in terms of new constraints or tasks or by retracting parts of the plan.

---

[23] The STOW-97 ACTD was designed to evaluate advanced distributed simulation (ADS) technology by linking a large number of simulated entities during a single, coordinated exercise.

All learning occurs through chunking, which is the acquisition of productions that occur during the process of impasse resolution and subgoaling. Chunk acquisition models task performance as evolving from the controlled, deliberative processing using weak methods to the automated processing using strong methods based on direct access to LTM.

Preliminary work has addressed the incorporation of emotional behaviors into Soar (Jones, Henninger, and Chown, 2002). This effort grafts a connectionist framework onto the symbolic Soar architecture. The connectionist framework is organized into three levels. At the lowest level are sources of confusion and clarity that arise from SA. These give rise to sensations of pain and pleasure that represent an interpretation of stimuli as either a threat or enhancement to survival. Pain and pleasure, in turn, give rise to arousal, which acts as the interface between the connectionist emotional system and the symbolic cognitive systems.

Whereas Soar was designed as a model of individual intelligence, systems such as TacAir-Soar comprise multiple interacting agents that emulate social systems. Laird, Jones, and Nielsen (1994) noted that in military organizations, such as those represented in TacAir Soar, coordination is highly constrained by specific tactics and doctrine and that communication is limited and proceduralized. Assuming these constraints, they incorporated into Soar some rules that reflected doctrine and tactics that govern real-time coordination in rotary-wing attack missions. Although they were able to simulate some aspects of coordination, the system was not able to cope with unexpected events, such as the loss of a key team member. Recognizing the essential inflexibility of such proceduralized coordination, Soar developers devised an explicit model of team goals and plans that are shared among team members. The resulting model, called a Shell for TEAMwork (STEAM), represents an integration of team with individual knowledge (Tambe, 1996). STEAM has been used to model coordination among team members in rotary-wing companies and has more recently been used as the underlying method for improving teamwork in the Information Science Institute Synthetic (ISIS) team[24] entered in RoboCup '97, an international competition to test multiagent systems using soccer as a simulation test bed (Tambe et al., 1999).

---

[24] Milind Tambe, Jafar Adibi, Yaser Al-Onaizan, Ali Erdem, Gal A. Kaminka, Stacy Marsella, Ion Muslea, Marcelo Tallis. A team from the Information Science Institute (ISI) at the University of Southern California (USC). ISI is involved in a broad spectrum of information processing research and in the development of advanced computer and communication technologies.

4. Applications

One of the explicit goals of the Soar research program is that the model be pushed to demonstrate its ability to represent a variety of intelligent behaviors (e.g., Rosenbloom et al., 1991). Initial applications for Soar (like those in ACT-R) were in toy tasks (e.g., puzzles and games such as Tower of Hanoi and Cryptarithmetic) that capture specific aspects of cognition. Applications then spread to more practical domains, such as knowledge-intensive problems in medical diagnosis (Neomycin-Soar) and software design (Designer-Soar). It also tackled learning in complex expert systems (R1-Soar).

Unlike ACT-R, Soar has not been used much to model academic learning and performance. Exceptions are models of mathematical skills and knowledge underlying subtraction (Rosenbloom et al., 1991) and concept acquisition (Miller and Laird, 1996). Also, Lewis (1993) developed a model of NL(natural language) comprehension (NL-Soar) to address some classic problems in psycholinguistics. Finally, the NL-Soar model has been used to as a practical Soar implementation for modeling communications in air-to-air combat tactics (Lehman, VanDyke, and Rubinoff, 1995) and in management of NASA tests (Nelson, Lehman, and John, 1994).

Some of the more extensive and complex examples of Soar models have been implemented in the context of military tasks. The previously described Soar/IFOR project developed HBR models to control fixed- and rotary-wing aircraft in realistic combat situations. The technology developed in Soar/IFOR has since been transitioned to Air Force demonstrations of advance distributed training (Roadrunner ’98 and Coyote ’98) and to the Navy’s Battle Force Tactical Trainer (BFTT).

5. Technical Considerations

a. Input and Input Aids

User input includes a specification of operators that are relevant to task performance. Users may wish to specify all details of task performance if the model is intended to emulate expert performance. Jones et al. (1999) used this approach in TacAir Soar for STOW-97, which allowed them to disable Soar’s learning capability thereby increasing overall system performance. On the other hand, users may elect to emulate learning and provide Soar only the operators that describe top-level goals and fundamental operators (i.e., the primitives) that apply to task performance.

Input is aided by Visual Soar, a development environment written in Java. Visual Soar comprises three key components:

1. **Operator window.** Operator window graphically displays the hierarchy of operators in the model under development and allows users to add operators, delete operators, and modify operator names.
2. **Rule editor.** Rule editor displays all system rules and permits full text editing of all fields in rules and provides syntactic and semantic checks.
3. **Data map.** Data map provides access to items in WM, given firing of specific rules, and lets users set value types and ranges of variables in items.

b. Model Output and Analysis Tools

Model output is the result of changes to WM caused by production firings. However, not every change to WM results in an external reaction. The only changes that results in an external reaction are those changes that affect output-link structures, which provide connections to a top-level state. The output link then causes some external event, such as some action in a simulator or the storage of data in a computer file, to occur.

No specific analysis tools currently exist for Soar. However, the interface could be used to output data files that could be analyzed by standard statistical packages.

c. Computer Language and Interfaces

The Soar software has undergone 8 major revisions in its 20 years of development. It was originally implemented in LISP but has since (as of Version 6) been ported to C. One version of Soar7 (7.3) and three versions of Soar8 (8.2, 8.3, and 8.4) are currently supported by the Artificial Intelligence Laboratory at the University of Michigan. Most of the versions are available for implementation on all major PC operating systems: Microsoft Windows, Apple Macintosh4, and UNIX. All versions can be downloaded from the University of Michigan Web site at http://ai.eecs.umich.edu/soar/software.html. These downloads are free.

Soar7 was the first version to implement Tcl/Tk as the interface for the model. Tcl (Tool Command Language) is an open-source scripting language, and Tk is a toolkit for creating graphical user interfaces. John Ousterhout created these two bundled utilities as freeware. Links and instructions to the appropriate version of Tcl/Tk are provided on the download pages for versions of Soar that are currently supported.

Version 8.4 of Soar is particularly notable because it no longer includes any interface code in the Soar kernel, thereby making a clear distinction between the kernel and interface. Soar8.4 also includes a C-level API. The API has interfacing functions similar to Tcl but provides increased capability to embed Soar in other C-based software applications.

6. Evaluation

To the extent that Soar has provided creditable models of performance in multiple complex simulations, it represents the state-of-the-art in HBR. Furthermore, its relatively simple architecture fits an impressive range of task domains. Whereas one might argue that ACT-R is more popular among academic users, Soar is more prevalent in actual operational applications.

Although Soar developers are enthusiastic supporter of the model, they are also mindful of its limitations. In his comparison of Soar and ACT-R, Young (1999) pointed out the following shortcomings:

- Soar does not provide precise quantitative predictions of performance time and error rates in the way that ACT-R does.
- The Soar modeler is encouraged to use only the basic set of operators and to avoid inventing any new capabilities. Soar developers have been accused of "listening to the model"—that is, avoid thinking how people perform a task and instead considering how Soar would do it.
- The simplicity of Soar leaves too much to the modeler's discretion—particularly when the task is far afield from Soar's original milieu (i.e., problem solving). Two Soar modelers could build two quite different but equally plausible models of the same process. In these instances, Soar may be, in fact, relatively unconstrained.

III. SUMMARY AND CONCLUSIONS

Table III-1 compares the HBR models with regard to the cognitive and behavioral functions they simulate. The table entries describe the outcome of dichotomous yes/no judgments of whether each model is capable of emulating the function in question. This presentation differs in style and intent from summary tables in the two previous reviews (Pew and Mavor, 1998; Ritter, Shadbolt, et al., 2002), where entries were short textual passages describing model capability with regard to the function. Given the present format, a few words of caution are appropriate in interpreting Table III-1:

- Reducing the description to yes/no judgments does not convey the quality and extent to which the model actually models a particular function. The reader should regard the entries as suggestive and consult the appropriate part of Section II for a more detailed description of model capabilities.
- To earn a "yes" judgment, the documentation and other literature associated with the model had to describe the model's capabilities specifically for that particular function. No inferences were made about the model's potential capabilities to emulate the function with modification. Model adherents may reasonably argue that the model can demonstrate additional functions with appropriate modifications.
- Judgments were based on the published documents that are provided in the reference section. Although our review of the literature was extensive, it was not completely exhaustive. References that were not unearthed may contain evidence for additional model capabilities.
- Despite appearances, Table III-1 does not represent a "scorecard" with which to rate the merits of HBR models. The fact that one model emulates 5 functions and another emulates 10 functions does not reflect their relative worth to model users. The match of model functions to the simulation requirement is what should matter to the user.

These cautionary statements notwithstanding, Table III-1 suggests some generalizations that one can make about the current state of the art in human behavior modeling:

- Decision-making is a universal function of all models. However, some of the models emulate only a reactive type of decision-making—that is, one that can be described by if-then type rules. This is not surprising in light of the

Table III-1. Summary of Cognitive and Behavioral Functions Represented in HBR Models Reviewed in the Present Study

	Cognitive Function Required											
Acronym/ Abbreviation	**Perception**	**Psychomotor Performance**	**Attention**	**Situation Awareness (SA)**	**Short-term Memory (STM)**	**Long-term Memory (LTM)**	**Learning**	**Decision-Making**	**Problem Solving**	**Cognitive Workload**	**Emotional Behavior**	**Social Behavior**
ACT	X	X	X		X	X	X	X	X			
ART	X		X		X	X	X	X				
APEX	X	X				X		X				
Brahms	X	X				X		X				X
CogAff	X	X			X	X		X			X	
COGNET	X	X	X	X	X			X	X	X		
CCT	X	X			X	X		X				
COGENT					X	X	X	X				
CAPS			X		X	X		X	X			
C-I Theory			X		X	X		X				
DCOG	X		X		X	X		X		X		
EPIC	X	X			X	X		X				
HOS	X	X	X		X			X				
MIDAS	X	X	X	X	X	X		X				X
Micro SAINT	X		X			X		X		X		
OMAR	X		X			X		X				X
PSI	X	X	X		X	X	X	X	X		X	
SAMPLE	X		X	X		X		X				X
Soar	X	X	X	X	X	X	X	X	X		X	X

Note for Table III-1: *An "X" entry indicates that the function is emulated by the model.*

fact that most of the HBR models can be classified as "rule-based." Even the few non-rule-based connectionist models (e.g., ART, CAPS, C-I) have the capability to emulate this basic form of decision-making.

- All the models have the capability to represent either STM or LTM. Again, most of the models are derived from the tradition of human information processing, which employs the tools and metaphors of computer science. Memory storage is a central concept to both disciplines.

- The "front-end" of cognition (perception and attention) is well represented in most models. Similarly, the ultimate output from cognition (psychomotor action) is modeled, at least to a minimal degree, in most models. The attention to inputs and outputs may also be a reflection of the information-processing tradition in human behavioral modeling.
- In contrast, learning and problem-solving functions are represented in relatively few models. Taken together with the apparent ability of models to emulate the "front-end" of cognition, this suggests the following: Whereas most HBR models may be good at *reacting* to expected situations (i.e., situations for which they are programmed), they may not be so good at *adapting* to novel situations.
- The capability to emulate SA is explicitly represented in only 4 of the 18 models. However, this deficiency may be more apparent than real. Given the capability of most models to emulate perception and attention, it may be the case that they can also represent SA functions. What is needed is an unambiguous and explicit definition of SA for HBR models.
- Very few of the models have the capability to simulate emotional or social behaviors. Recent publications and presentations suggest that these may be "growth areas" for current and future HBR models. Although not reviewed in this document, a related growth area is the representation of personality or dispositional factors in HBR models.

To go beyond simply describing the models to recommending models for particular applications requires an additional step: specifying the link between these cognitive/behavioral functions and human simulation roles that HBR models potentially fill in military simulations. Enumerating all the possible roles that an HBR model might emulate would be difficult, but Table III-2 lists five representative roles: equipment operators, maintenance technicians, warfighters, small teams and crews, and commanders and staffs. The table specifies the cognitive/behavioral functions that each of those roles require. Although the table is highly speculative, two inferences can be drawn:

1. Only more sophisticated models (e.g., ACT-R, Soar) are appropriate for higher echelon roles, but less sophisticated models (HOS, COGNET, EPIC) may be sufficient for lower echelon. Although the more complex models also fit the lower echelon roles, the less complex models have the advantage of being less computationally intensive, which is an important consideration in M&S.
2. None of the models fully satisfy all requirements, particularly those roles requiring complex interactions among entities. It is also probable that the

Table III-2. Summary of Cognitive and Behavioral Functions Required by Roles T hat HBR Models Could Assume in Military Simulations

	Required Cognitive/Behavioral Functions											
Military Role	**Perception**	**Psychomotor Performance**	**Attention**	**Situation Awareness (SA)**	**Short-term Memory (STM)**	**Long-term Memory (LTM)**	**Learning**	**Decision-Making**	**Problem Solving**	**Cognitive Workload**	**Emotional Behavior**	**Social Behavior**
Equipment Operators	X	X	X		X	X		X		X		
Maintenance Technicians	X	X	X		X	X		X	X			
Warfighters	X	X	X	X	X	X		X	X	X	X	
Small Teams and Crews	X	X	X	X	X	X		X	X	X	X	X
Commanders and Staffs	X	X	X	X	X	X	X	X	X	X	X	X

Note for Table III-2: *An "X" entry specifies the cognitive/behavioral functions that each of those roles require.*

table identifies all cognitive/behavioral functions that are required. The point is that HBR models have plenty of conceptual problems to address in future versions.

In conclusion, the field of HBR modeling appears even riper for application to military simulations than it seemed in previous reviews (Pew and Mavor, 1998; Ritter, Shadbolt, et al., 2002), because of the proliferation of HBR models and of the impressive demonstrations of these models interacting with military and military-like simulations. However, the potential HBR user must consider some of the serious impediments to such applications.

Foremost among these impediments is the level of expertise required to interface an HBR with a military simulation. This expertise includes the conceptual understanding of the HBR model as a valid representation of cognition and behavior, the technical knowledge required to control inputs to and outputs from the military simulation in question, and the computer science expertise required to understand and make appropriate modifications to code in the HBR model and in the military simulation. To state this more

plainly, none of the models is remotely close to providing "plug and play" interoperability with a military simulation. For HBR models to have an impact on military simulation and training, the fundamental issue of the interoperability of HBR models and simulations must be addressed.

REFERENCES

Abrett, G., Burstein, M., and Deutsch, S. (1989). *An environment for building goal-directed, knowledge-based simulations* (BBN Technical Report 7062). Cambridge, MA: BBN Corporation.

Acquisti, A., Clancey, W.J., van Hoof, R., Scott, M., and Sierhuis, M. (2001, December). *Brahms tutorial* (Technical Memorandum TM01-0002, Version 0.9.9.4 RFC). Moffett Field, CA: NASA Ames Research Center. Retrieved September 25, 2002, from AgentiSolutions Web site: http://www.agentisolutions.com/documentation/tutorial/Brahms_Tutorial.pdf.

Agha, G.A. (1986). *Actors: A model of concurrent computation in distributed systems*. MIT Press, Cambridge, MA, 1986.

Anderson, J.R. (1976). *Language, memory, and thought*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Anderson, J.R. (1983). *The architecture of cognition*. Cambridge, MA: Harvard University Press.

Anderson, J.R. (1990). *The adaptive character of thought*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Anderson, J.R. (1993). *Rules of the mind*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Anderson, J.R., and Bower, G.H. (1973). *Human associative memory*. Washington, DC: V.H. Winston and Sons.

Anderson, J.R., and Lebiere, C. (1999). *The atomic components of thought*. Mahwah, NJ: Lawrence Erlbaum Associates.

Anderson, J.R., Bothell, D., Byrne, M.D., and Lebiere, C. (2002, September). *An integrated theory of the mind*. [Manuscript submitted for publication]. Retrieved March 25, 2003, from ACT-R Web site: http://act-r.psy.cmu.edu/papers/403/IntegratedTheory.pdf.

Aptima, Inc. (2001). *Compute-generated forces for team training* [Web page]. Retrieved May 20, 2002, from Aptima Web site: http://www.aptima.com/Projects/Computer_Generated_Forces.html.

Atkinson, R.C., and Shiffrin, R.M. (1968). Human memory: A proposed system and its control processes. In K.W. Spence and J.T. Spence (Eds.), *The psychology of learning and motivation*. New York: Academic Press.

Aykroyd, P., Harper, K.A., Middleton, V., and Hennon, C.G. (2002). Cognitive modeling of individual combatant and small unit decision-making within the Integrated Unite Simulation System. In *Proceedings of the 11th Conference on Computer Generated Forces and Behavior Representation*. Orlando, FL: Simulation Interoperability Standards Organization.

Baron, S., Muralidharan, R., Lancrafty, R., and Zacharias, G. (1980). PROCRU: A model for analyzing crew procedures in approach to landing (BBN Technical Report No. 4374). Cambridge, MA: Bolt, Beranek, and Newman, Inc.

Bartl, C., and Dörner, D. (1998). *Comparing the behaviour of PSI with human behaviour in the BioLab game* (Memorandum Number 32). Bamberg, Germany: Universität Bamberg: Lehrstuhl Psychologie II.

Blackmon, M.H., Polson, P.G., Kitajima, M., and Lewis, C. (2002). Cognitive walkthrough for the web. In *Proceedings of CHI 2002, ACM Conference on Human Factors in Computing Systems, CHI Letters*, *4(1)*, 463–470.

Bovair, S., and Kieras, D.E. (1984). A guide to propositional analysis for research on technical prose. In B.K. Britton and J.B. Black (Eds.), *Understanding expository text* (pp. 315–362). Hillsdale, NJ: Lawrence Erlbaum Associates.

Bovair, S., Kieras, D.E., and Polson, P.G. (1990). The acquisition and performance of text-editing skill: A cognitive complexity analysis. *Human Computer Interaction*, *5*, 1–48.

Bradshaw, J.M., Sierhuis, M., Gawdiak, Y., Jeffers, R., Suri, N., and Greaves, M. (2001). Adjustable autonomy and teamwork for the personal satellite assistant. In Bernhard Nebel (Ed.), *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*. San Francisco, CA: Morgan Kaufman.

Bratman, M.E. (1987). *Intentions, plans, and practical reason*. Cambridge, MA: Harvard University Press.

Brown, M.J., Jr. (2000). Tools for human performance modeling and simulation in support of command and control analysis. In *Proceedings of the 5th International Command and Control Research and Technology Symposium*. Canberra, Australia.

Byrne, M.D. (1994). Integrating, not debating, situated action and computational models: Taking the environment seriously. In A. Ram and K. Eiselt (Eds.), *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society* (pp. 118–123). Hillsdale, NJ: Lawrence Erlbaum Associates.

Byrne, M.D. (2002). Cognitive architectures. In J.A. Jacko and A. Sears (Eds.), *Handbook of human-computer interaction*. Mahwah, NJ: Lawrence Erlbaum Associates.

Card, S.K., Moran, T.P., and Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Carley, K., and Wendt, K. (1991). Electronic mail and scientific communication. *Knowledge: Creation, Diffusion, Utilization*, *12*, 406–440.

Carpenter, G.A., and Grossberg, S. (1987a). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, *37*, 54–115.

Carpenter, G.A., and Grossberg, S. (1987b). ART2: Self-organizing of stable category recognition codes for analog input patterns. *Applied Optics*, *26*, 4919–4930.

Carpenter, G.A., and Grossberg, S. (1987c). ART3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures. *Neural Networks*, *3*, 129–152.

Carpenter, G.A., Grossberg, S., and Reynolds, J.H. (1991). ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Networks*, *4*, 565–588.

Chong, R.S., and Laird, J.E. (1997). Identifying dual-task executive process knowledge using EPIC-Soar. In *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society* (pp. 107–112). Hillsdale, NJ: Lawrence Erlbaum Associates.

Clancey, W.J. (1993). Situated action: A neurophysiological interpretation (response to Vera and Simon). *Cognitive Science, 17,* 87–107.

Clancey, W.J. (1997). *Situated cognition: On human knowledge and computer representations*. Cambridge: Cambridge University Press.

Clancey, W.J. (2002). Simulating activities: Relating motives, deliberation, and attentive coordination. *Cognitive Systems Research*, *3*, 471–499.

Clancey, W.J., and Malin, J. (2002, January). Brahms-CONFIG: Integrated simulation of habitat work practices and systems. Paper presented at the Second Biennial Space Human Factors Workshop, Center for Advanced Space Studies, Houston, TX.

Clancey, W.J., Sachs, P., Sierhuis, M., and van Hoof, R. (1998). Brahms: Simulating practice for work systems design. *International Journal of Human-Computer Studies*, *49*, 831–865.

Computer-Human Interaction Laboratory (CHIL) (2002, May). [ACT-R/PM Web page.] Retrieved March 25, 2003, from CHIL Rice University Web site: http://chil.rice.edu/byrne/RPM/project.html.

Cooper, R. (1995). Towards an object-oriented language for cognitive modeling. In J.D. Moore and J.F. Lehman (Eds.), *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society* (pp. 556–561). Hillsdale, NJ: Lawrence Erlbaum Associates.

Cooper, R., and Fox, J. (1997). Learning to make decisions under uncertainty: The contributions of qualitative reasoning. In M.G. Shafto and P. Langley (Eds.), *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society* (pp. 125–130). Mahwah, NJ: Lawrence Erlbaum Associates.

Cooper, R., Yule, P., and Sutton, D. (1998). COGENT: An environment for the development of cognitive models. In U. Schmid, J.F. Krems, and F. Wysotzki (Eds.), *A cognitive science approach to reasoning, learning, and discovery* (pp. 55–82). Lengerich, Germany: Pabst Science Publishers.

Corker, K.M. (2001). *Software* [Web Page]. Retrieved May 2, 2002 from HAIL Web site: http://www.engr.sjsu.edu/hfe/hail/software.htm.

Corker, K.M., and Smith, B.R. (1993). An architecture and model for cognitive engineering simulation analysis: Application to advanced aviation automation. In *Proceedings of the AIAA Computing in Aerospace 9 Conference*. Santa Monica, CA: American Institute of Aeronautics and Astronautics.

Cramer, N. (1998). Distributed-OMAR: Reconfiguring a LISP system as a hybrid LISP/(Java) component. Paper presented at the *LISP Users Group Meeting*. Berkeley, CA: Association of LISP Users.

Deutsch, S. (1997). Multi-agent human performance modeling in OMAR. In M. Smith, G. Slavendy, and R. Koubek (Eds.), *Design of computing systems: Social and ergonomic considerations* (pp. 79–82). Amsterdam: Elsevier.

Deutsch, S. (1998). Multi-disciplinary foundations for multiple-task human performance modeling in OMAR. Paper presented at the *Twentieth Annual Meeting of the Cognitive Science Society*. Madison, WI: Cognitive Science Society.

Deutsch, S. (2002). *D-OMAR: Distributed Operator Model Architecture* [Web page]. Retrieved May 17, 2002, from BBN Technologies Web site: http://omar.bbn.com/index.html.

Deutsch, S., and Adams, M.J. (1995). The operator-model architecture and its psychological framework. Paper presented at the *6th IFAC Symposium on Man-Machine Systems*. Cambridge, MA: Massachusetts Institute of Technology.

Deutsch, S.E., Adams, M.J., Abrett, G.A., Cramer, N.L., and Feehrer, C.E. (1993). *Operator model architecture: Software functional specification*. (AI/I-IR-TP-1993-0027) Wright-Patterson AFB, OH: Armstrong Laboratory, Logistics Research Division.

Deutsch, S., and Benyo, B. (2001). The D-OMAR simulation environment for the AMBR experiments. In *Proceedings of the 10th Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL: Simulation Interoperability Standards Organization.

Deutsch, S.E., MacMillian, J., and Cramer, N.L. (1993). *Operator Model Architecture (OMAR) demonstration final report* (AL/HR-TR-1996-0161). Wright-Patterson AFB, OH: Armstrong Laboratory, Logistics Research Division.

Deutsch, S., MacMillan, J., Cramer, N., and Chopra, S. (1997). *Operator Model Architecture (OMAR) support* (BBN Technical Report 8179). Cambridge, MA: Bolt, Beranek, and Newman.

Diller, D., and Tenney, Y. (2002). Experiment design and comparison of human and model data. In K.A. Gluck and R.W. Pew (Chairs), The AMBR model comparison project: Round III—Modeling category learning. In W.D. Gray and C. Shunn (Eds.), *Proceedings of the Twenty-Fourth Annual Conference of the Cognitive Science Society* (p. 21). Mahwah, NJ: Lawrence Erlbaum Associates.

Dörner, D., and Hille, K. (1995). Artificial souls: Motivated and emotional robots. In *Proceedings of the International Conference on Systems, Man, and Cybernetics* (Volume 4, pp. 3828–3832). Piscataway, NJ: IEEE.

Eggleston, R.G., Young, M.J., and McCreight, K.L. (2000). Distributed cognition: A new type of human performance model. In M. Freed (Ed.), *Simulating human agents: Papers from the 2000 AAAI Fall Symposium* (Technical Report FS-00-03) (pp. 8–14). Menlo Park, CA: AAAI Press.

Eggleston, R.G., Young, M.J., and McCreight, K.L. (2001). Modeling human work through distributed cognition. In *Proceedings of the 10th Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL: Simulation Interoperability Standards Organization.

Emery, F.E., and Trist, E.L. (1960). Socio-technical systems. In C.W. Churchman and M. Verhuilst (Eds.), *Management sciences: Models and techniques* (Vol. 2). London: Pergamon.

Emmerson, P., and Nibbelke, R. (2000). iGEN™ [Software Review]. *Ergonomics in Design, Summer,* 29–31.

Endsley, M.R. (1988). Design and evaluation for situation awareness enhancement. In *Proceedings of the 32nd Annual Meeting of the Human Factors Society* (pp. 97–101). Santa Monica, CA: Human Factors Society.

Ericsson, K.A., and Kintsch, W. (1995). Long-term working memory. *Psychological Review*, *102*, 211–245.

Firby, R. J. (1989). *Adaptive execution in complex dynamic worlds*. Unpublished doctoral dissertation, Yale University, New Haven, CT.

Freed, M.A. (1998). *Simulating human performance in complex, dynamic environments*. Unpublished doctoral dissertation, Northwestern University, Evanston, IL.

Freed, M., and Remington, R. (2000). GOMS, GOMS+, and PDL. In Working Notes of the *AAAI Fall Symposium on Simulating Human Agents*. Falmouth, MA.

Freed, M., Dahlman, E., Dalal, M., and Harris, R. (2002). *Apex reference manual for Apex version 2.2*. Moffett Field, CA: NASA ARC.

Freeman, B. (2002). *D-OMAR: OmarL User/Programmer Manual* (Version 4.0). [Online document]. Retrieved May 20, 2002, from BBN Technologies OMAR Web site: http://omar.bbn.com/manual/index.html.

Freeman, J.T., Pharmer, J.A., Lorenzen, C., Santoro, T.P., and Kieras, D. (2002, June). Complementary methods of modeling team performance. Paper presented at the 2002 Command and Control Research and Technology Symposium. Monterey, CA.

Glenn, F.A., and Doane, S.M. (1981). *A human operator simulator model of the NASA terminally configured vehicle (TCV).* (Contract No. NAS1-15983) Washington, DC: NASA.

Glenn, F., Schwartz, S., and Ross, L. (1992). *Development of a Human Operator Simulator Version V (HOS-V): Design and implementation* (Research Note 92-PERI-POX). Alexandria, VA: U.S. Army Research Institute for the Behavioral and Social Sciences.

Gluck, K.A., and Pew, R.W. (Chairs) (2002). The AMBR model comparison project: Round III—Modeling category learning. In W.D. Gray and C. Shunn (Eds.), *Proceedings of the Twenty-Fourth Annual Conference of the Cognitive Science Society* (p. 21). Mahwah, NJ: Lawrence Erlbaum Associates.

Goel, V., Pullara, S.D., and Grafman, J. (2001). A computational model of frontal lobe dysfunction: Working memory and the Tower of Hanoi task. *Cognitive Science*, *25*, 287–313.

Gong, R., and Kieras, D. (1994). A validation of the GOMS model methodology in the development of a specialized, commercial software application. In *Proceedings of CHI94* (pp. 351–357). New York: Association of Computing Machinery.

Gore, B.F., and Corker, K.M., (1999). *System interaction in free flight: A modeling tool cross-comparison* (SAE Technical Paper 1999-01-1897). Warrendale, PA: The Society of Automotive Engineers.

Green, P. (1999). *Navigation system data entry: Estimation of task times* (UMTRI 99-17). Ann Arbor, MI: The University of Michigan Transportation Research Institute.

Grossberg, S. (1976a). Adaptive pattern classification and universal recoding, I: Parallel development and coding of neural feature detectors. *Biological Cybernetics*, *23*, 121–134.

Grossberg, S. (1976b). Adaptive pattern classification and universal recoding, II: Feedback, expectation, olfaction, and illusions. *Biological Cybernetics*, *23*, 187–202.

Grossberg, S. (1995). The attentive brain. *American Scientist*, *83*, 438–449.

Grossberg, S. (1999). The link between brain learning, attention, and consciousness. *Consciousness and Cognition*, *8*, 1–44.

Grossberg, S. (2000). Linking mind to brain: The mathematics of biological intelligence. *Notices of the American Mathematical Society*, *47*, 1361–1372.

Grossberg, S., and Gutowski, W.E. (1987). Neural dynamics of decision-making under risk: Affective balance and cognitive-emotional interactions. *Psychological Review*, *94*, 300–318.

Hanson, M.L., Harper, K.A., Endsley, M., and Rezsonya, L. (2002). Developing cognitively congruent HBR models via SAMPLE: A case study in airline operations modeling. In *Proceedings of the 11th Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL: Simulation Interoperability Standards Organization.

Hanson, M.L., Sullivan, O., and Harper, K.A. (2001). On-line situation assessment for unmanned air vehicles. In I. Russell and J. Kolen (Eds.), *Proceedings of the Fourteenth International FLAIRS Conference*. Menlo Park, CA: AAAI Press

Harper, K.A., Ho, S.S., Zacharias, G.L., and Raibert, M. (2000). Intelligent hostile urban threat agents for MOUT operations. In *Proceedings of the 9th Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL: Simulation Interoperability Standards Organization.

Harper, K.A., Mulgund, S.S., Zacharias, G.L., and Kuchar, J.K. (1998). Agent-based performance assessment tool for general aviation operations under free flight. In *AIAA Guidance, Navigation, and Control Conference* (Vol. 1, pp. 1–10). Boston, MA: American Institute of Aeronautics and Astronautics, Inc.

Harper, K.A., Ton, N., Jacobs, K., Hess, J., and Zacharias, G.L. (2001). Graphical agent development environment for human behavior representation. In *Proceedings of the 10th Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL: Simulation Interoperability Standards Organization.

Harper, K.A., and Zacharias, G.L. (2002). Modeling attention allocation and multitasking in computational human behavior representations. In *Proceedings of the 11th Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL: Simulation Interoperability Standards Organization.

Harris, R., Iavecchia, H.P., and Dick, A.O. (1989). The Human Operator Simulator (HOS-IV). In G.R. McMillan, D. Beevis, E. Salas, M.H. Strub, R. Sutton, and L Van Breda (Eds.), *applications of human performance models to system design* (pp. 275–280). New York: Plenum.

Hart, S.G., Dahn, D., Atencio, A., and Dalal, K.M. (2001). *Evaluation and application of MIDAS v2.0* (SAE Technical Paper 2001-01-2648). Warrendale, PA: The Society of Automotive Engineers.

Hegarty, M. (2001, May). *Capacity limits in mechanical reasoning*. Paper presented at the Fifteenth International Workshop on Qualitative Reasoning, San Antonio, TX.

Hicinbotham, J.H. (2001). Maintaining situation awareness in synthetic team members. In *Proceedings of the 10th Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL: Simulation Interoperability Standards Organization.

Hill, R.W., Jr. (1999). Modeling perceptual attention in virtual humans. In *Proceedings of the 8th Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL: Simulation Interoperability Standards Organization.

Hill, R., Chen, J., Gratch, J., Rosenbloom, P. and Tambe, M. (1998). Soar-RWA: Planning, teamwork, and intelligent behavior for synthetic rotary-wing aircraft. In *Proceedings of the 7th Conference on Computer Generated Forces and Behavioral Representation*. Orland, FL: Simulation Interoperability Standards Organization.

John, B.E., and Kieras, D.E. (1996). The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction*, *3*, 320–351.

John, B., Vera, A., Matessa, M., Freed, M., and Remington, R. (2002). Automating CPM-GOMS. *CHI Letters*, *4*, 147–154.

Jones, R.M., Henninger, A.E., and Chown, E. (2002). Interfacing emotional behavior moderators with intelligent synthetic forces. In *Proceedings of the 11th Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL: Simulation Interoperability Standards Organization.

Jones. R.M., Laird, J.E., Nielsen, P.E., Coulter, K.J., Kenny, P., and Koss, F.V. (1999). Automated intelligent pilots for combat flight simulation. *AI Magazine*, *20*, 27–41.

Just, M.A., and Carpenter, P.A. (1992). A capacity theory of comprehension: Individual differences in working memory. *Psychological Review*, *99*, 122–149.

Just, M.A., Carpenter, P.A., Keller, T.A., Eddy, W.F., and Thulborn, K.R. (1996). Brain activation modulated by sentence comprehension. *Science*, *274*, 114–116.

Just, M.A., Carpenter, P.A., and Varma, S. (1999). Computational modeling of high-level cognition and brain function. *Human Brain Mapping*, *8*, 128–136.

Kieras, D.E. (1994). *A guide to GOMS task analysis*. Unpublished manuscript, University of Michigan.

Kieras, D.E. (1996). *A guide to GOMS model usability evaluation using GOMSL* [On-line publication]. Retrieved August 27, 2002, from the University of Michigan's Electrical Engineering and Computer Science Department Web site: http://www.eecs.umich.edu/people/kieras/GOMS/GOMSL_Guide.pdf.

Kieras, D.E. (1999). *A guide to GOMS model usability evaluation using GOMSL and GLEAN3* [On-line publication]. Retrieved August 27, 2002, from the University of Michigan's Electrical Engineering and Computer Science Department Web site: http://www.eecs.umich.edu/people/kieras/GOMS/GOMSL_Guide.pdf.

Kieras, D.E., and Meyer, D.E. (1995). *An overview of the EPIC architecture for cognition and performance with application to human-computer interaction* (EPIC Report No. 5). Ann Arbor, MI: The University of Michigan.

Kieras, D.E., and Meyer, D.E. (1998, May). *The EPIC architecture: Principles of operation* [On-line document]. Retrieved April 1, 2003, from the University of Michigan's EPIC Web site: http://www.eecs.umich.edu/~kieras/epic.html.

Kieras, D.E., and Polson, P.G. (1985). An approach to the formal analysis of user complexity. *International Journal of Man-Machine Studies*, *22*, 365–394.

Kieras, D.E., Wood, S.D., Abotel, K., and Hornof, A. (1995). GLEAN: A computer-based tool for rapid GOMS model usability evaluation of user interface designs. In *Proceedings of the Symposium on User Interface Software and Technology (UIST 95)* (pp. 91–100). New York: Association of Computing Machinery.

Kintsch, W. (1988). The role of knowledge in discourse comprehension: A construction-integration model. *Psychological Review, 95,* 163–182.

Kintsch, W. (1998) *Comprehension: A paradigm for cognition*. New York: Cambridge University Press.

Kintsch, W. (2002). Predication. *Cognitive Science*, *25*, 173–202.

Kintsch, W., and van Dijk, T.A. (1978). Toward a model of text comprehension and production. *Psychological Review*, *85*, 363–394.

Kintsch, W., and Vipond, D. (1978). Reading comprehension and readability in educational practice and psychological theory. In L.G. Nillson (Ed.), *Memory: Processes and problems*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Kintsch, W., Patel, V.L., and Ericsson, K.A. (1999). The role of long-term working memory in text comprehension. *Psychologia*, *42*, 186–198.

Kitajima, M., and Polson, P.G. (1995). A comprehension-based model of correct performance and errors in skilled, display-based, human-computer interaction. *International Journal of Human-Computer Studies*, *43*, 65–99.

Kitajima, M., and Polson, P.G. (1997). A comprehension-based model of exploration. *Human-Compute Interaction*, *12*, 345–389.

Kitajima, M., and Polson, P.G. (1998). Knowledge required for understanding task-oriented instructions. In J. Tanaka (Ed.), *Proceedings of the Third Asian Pacific Computer-Human Interaction Conference (APCHI '98)* (pp. 19–24). Los Alamitos, CA: IEEE Computer Society.

Kitajima, M., Blackmon, M.H., and Polson, P.G. (2000). A comprehension-based model of web navigation and its application to web usability analysis. In S. McDonald, Y. Waern, and G. Cockton (Eds.), *People and Computers XIV-Usability or Else!* [Proceedings of HCI 2000] (pp. 357–373). Heidelberg, Germany: Springer-Verlag.

Kitajima, M., Soto, R., and Polson, P.G. (1998). LICAI+: A comprehension-based model of the recall of action sequences. In F.E. Ritter and R.M. Young (Eds.), *Proceedings of the 2nd European Conference on Cognitive Modeling* (pp. 82–89). Thrumpton, Nottingham: Nottingham University Press.

Klein, G.A. (1989). Recognition-primed decisions. In W. Rouse (Ed.), *Advances in man-machine systems research, Volume 5* (pp. 47-92). Greenwich, CT: JAI Press.

Kleinman, D., Baron, S., and Levison, W. (1971). A control theoretic approach to manned-vehicle systems. *IEEE Transactions on Automatic Control*, *No. 61*, 824–832.

Krafft, M.F. (2002, October). *Adaptive resonance theory*. Retrieved March 5, 2003, from the University of Zurich, Department of Information Technology Web site: http://www.ifi.unizh.ch/staff/krafft/papers/2001/wayfinding/html/node97.html.

Laird, J.E., and Newell, A. (1983). A universal weak method: Summary of results. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence* (pp. 771–773). Los Altos, CA: Kaufman.

Laird, J.E., and Rosenbloom, P.S. (1994). *The evolution of the Soar cognitive architecture* (Technical Report CSE-TR-219-49). Ann Arbor, MI: Department of Electrical Engineering and Computer Science, University of Michigan.

Laird, J.E., Congdon, C.B., and Coulter, K.J. (1999). *The Soar user's manual: Version 8.2*. Retrieved June 4, 2000, from a University of Michigan Web site: http://ai.eecs.umich.edu/soar/docs/manuals/soar8manual.pdf.

Laird, J.E., Jones, R.M., and Nielsen, P.E. (1994). Coordinated behavior of computer generated forces in TacAir-Soar. In *Proceedings of the 4th Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL: Simulation Interoperability Standards Organization.

Landauer, T.K. (1998). Learning and representing verbal meaning: The latent semantic analysis theory. *Current Directions in Psychological Science*, *7*, 161–164.

Landauer, T.K., and Dumais, S.T. (1997). A solution to Plato's Problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, *104*, 211–240.

Landauer, T.K., Foltz, P.W., and Laham, D. (1998). An introduction to latent semantic analysis. *Discourse Processes*, *25*, 259–284.

Landauer, T.K., Laham, D., Rehder, R., and Schreiner, M.E. (1997). How well can passage meaning be derived without using word order? A comparison of Latent Semantic Analysis and humans. In M.G. Shafto and P. Langley (Eds.), *Proceedings of the 19th Annual Meeting of the Cognitive Science Society* (pp. 412–417). Mahwah, NJ: Lawrence Erlbaum Associates.

Laughery, K.R., Jr., and Corker, K. (1997). Computer modeling and simulation. In G. Salvendy (Ed.), *Handbook of human factors and ergonomics (2nd ed.)* (pp. 1375–1408). New York: John Wiley & Sons.

LaVine, N., Kehlet, R., O'Connor, M.J., and Jones, D.L. (1999). Transferring ownership of ModSAF behavioral attributes. In *Proceedings of the Spring Simulation Interoperability Workshop*. Orlando, FL: Simulation Interoperability Standards Organization.

Lebiere, C. (2002). *Introduction to ACT-R 5.0* [tutorial]. Presented at *Twenty-Fourth Annual Conference of the Cognitive Science Society*, Fairfax, VA.

Le Mentec, J-C., Glenn, F., Zachary, W., Eilbert, J. (1999). Representing human sensory and motor action behavior in a cognitive modeling architecture. In *Proceedings of the 8th Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL: Simulation Interoperability Standards Organization.

Lehman, J.F., VanDyke, J., and Rubinoff, R. (1995). Natural language processing for IFORs: Comprehension and generation in the air combat domain. In *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL: Simulation Interoperability Standards Organization.

Leont'ev, A.N. (1979). The problem of activity in psychology. In J.V. Wertsch (Ed.), *The concept of activity in Soviet psychology* (pp. 37–71). Armonk, NY: M.E. Sharpe.

Lewis, R.L. (1993). *An architecturally based theory of human sentence comprehension*. Unpublished doctoral dissertation, Carnegie Mellon University, Pittsburgh, PA.

Lewis, R.L. (2001). Cognitive theory, Soar. In N.J. Smelser and P.B. Baltes (Eds.), *International encyclopedia of the social and behavioral sciences*. Amsterdam: Pergamon (Elsevier Science).

Linden, L.A. (1995). The ART Gallery documentation (V 1.0) [HTML document]. Retrieved March 26, 2003, from Boston University, Department of Cognitive and Neural Sciences and the Center for Adaptive Systems Web site: http://cns-web.bu.edu/pub/laliden/WWW/nnet_doc.html.

Linden, L. (n.d.). *The ART Gallery: A neural network simulation package*. Retrieved March 21, 2003, from the Boston University, Department of Cognitive and Neural Sciences Web site: http://cns-web.bu.edu/pub/laliden/WWW/nnet.html.

MacMillan, J., Deutsch, S.E., and Young, M.J. (1997). A comparison of alternatives for automated decision support in a multi-tasking environment. In *Proceedings for the 41st Annual Meeting of the Human Factors and Ergonomics Society*. Santa Monica, CA: Human Factors Society.

McCulloch, W.S., and Pitts, W. (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics, 5,* 115–133.

Meyer, D.E., and Kieras, D.E. (1997a). A computational theory of executive cognitive processes and multiple-task performance: Part 1. Basic mechanisms. *Psychological Review*, *104*, 3–65.

Meyer, D.E., and Kieras, D.E. (1997b). A computational theory of executive cognitive processes and multiple-task performance: Part 2. Accounts of psychological refractory-period phenomena. *Psychological Review*, *104*, 749–791.

Miller, C.S., and Laird, J.E. (1996). Accounting for graded performance within a discrete search framework. *Cognitive Science*, *20*, 499–537.

Mross, E.F., and Roberts, J.O. (1992). The construction-integration model: A program and manual (Publication No. 92-14). Bolder, CO: Institute of Cognitive Science, University of Colorado.

Mulgund, S.S., Harper, K.A., Zacharias, G.L., and Menke, T.E. (2000). SAMPLE: Situation Awareness Model for Pilot-in-the-Loop Evaluation. In *Proceedings of the 9th Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL: Simulation Interoperability Standards Organization.

Neisser, U. (1967). *Cognitive psychology*. Englewood Cliffs, NJ: Prentice-Hall.

Neisser, U. (1976). *Cognition and reality: Principles and implications of cognitive psychology*. San Francisco, CA: W.H. Freeman and Company.

Nelson, G., Lehman, J.F., and John, B. (1994). Integrating cognitive capabilities in a real-time task. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society* (pp. 658–663). Hillsdale, NJ: Lawrence Erlbaum Associates.

Newell, A. (1973). You can't play 20 questions with nature and win: Projective comments on the papers of this symposium. In W.G. Chase (Ed.), *Visual information processing* (pp. 283–310). New York: Academic Press.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Newell, A., and Simon, H.A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Newell, A., and Simon, H.A. (1956). The logic theory machine: A complex information-processing system. *IRE Transactions in Information Theory*, *IT-2*, 61–79.

Newell, A., Shaw, J.C., and Simon, H.A. (1957). Empirical explorations of the logic theory machine: A case study in heuristics. In *Proceedings of the 1957 Western Joint Computer Conference* (pp. 218–230). Also reprinted in E.A. Feigenbaum and J. Feldman (Eds.), *Computers and thought* (pp. 109–133). New York: McGraw-Hill, 1963.

Newell, A., Shaw, J.C., and Simon, H.A. (1958). Elements of a theory of human problem solving. *Psychological Review*, *65*, 151–166.

Pew, R.W., and Mavor, A.S. (Eds.) (1998). *Modeling human and organizational behavior: Applications to military simulations*. Washington, DC: National Academy Press.

Polson, P.G., Muncher, E., and Engelbeck, G. (1986). A test of common elements theory of transfer. In M. Mantei and P. Orbeton (Eds.), *Proceedings of the CHI '86 Conference on Human Factors in Computing* (pp. 78–83). New York: Association of Computing Machinery.

Porto, V.W., Fogel, D.B., and Fogel, L.J. (1998). Generating novel tactics through evolutionary computation. *SIGART Bulletin, Fall,* 8–14.

Pritsker, A.B., Wortman, D.B., Seum, C., Chubb, G., and Seifert, D.J. (1974). *SAINT: Systems Analysis of an Integrated Network of Tasks* (Aerospace Medical Research Laboratory, Technical Report AMRL-TR-73-126). Dayton, OH: Wright-Patterson Air Force Base.

Rao, A.S., and Georgeff, M.P. (1995). BDI agents: From theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)* (pp. 312–319). Menlo Park, CA: American Association for Artificial Intelligence.

Rasmussen, J. (1983). Skills, rules, and knowledge: Signals, signs and symbols, and other distinctions in human performance models. *IEEE Transactions On Systems, Man, and Cybernetics*, *SMC-13*, 257–266.

Ritter, F.E., Baxter, G.D., Avaramides, M., and Wood, A.B. (2002*). Soar: Frequently asked questions list* [Web page]. Retrieved May 29, 2002, from the Pennsylvania State University's Soar Web site:
http://ritter.ist.psu.edu/soar-faq/soar-faq.html.

Ritter, F.E., Shadbolt, N R., Elliman, D., Young, R., Gobet, F., and Baxter, G.D. (2002). *Techniques for modeling human and organizational behaviour in synthetic environments: A supplementary review*. Wright-Patterson Air Force Base, OH: Human Systems Information Analysis Center.

Rosenbloom, P.S. (1994). *A brief history of the Soar Project*. Downloaded May 30, 2002, from Soar Web site at Carnegie Mellon University: http://www-2.cs.cmu.edu/afs/cs/project/soar/public/www/brief-history.html.

Rosenbloom, P.S., Laird, J.E., Newell, A., and McCarl, R. (1991). A preliminary analysis of the Soar architecture as a basis for general intelligence. *Artificial Intelligence*, *47*, 289–325.

Ryder, J.M., Weiland, M.Z., Szczepkowski, M.A., and Zachary, W.W. (1998). Cognitive engineering of a new telephone operator workstation using COGNET. *International Journal of Industrial Ergonomics*, *22*, 417–429.

Ryder, J.M., and Zachary, W. (1991). Experimental validation of the attention switching component of the COGNET framework. In *Proceedings of the 35th Annual Meeting of the Human Factors Society* (pp. 72–76). Santa Monica, CA: Human Factors Society.

Salvucci, D.D., Boer, E.R., and Liu, A. (2001). Toward an integrated model of driver behavior in a cognitive architecture. *Transportation Research Record*, *1779*, 9–16.

Seamster, T.L., Redding, R.E., Cannon, J.R., Ryder, J.M., and Purcell, J.A. (1993). Cognitive task analysis of expertise in air traffic control. *International Journal of Aviation Psychology*, *3*, 257–283.

Siegel, A.I., and Wolf, J.J. (1962). A model for digital simulation of two-operator man-machine systems. *Ergonomics*, *5*, 557–572.

Siegel, A.I., and Wolf, J.J. (1969). *Man-machine simulation models*. New York: John Wiley.

Sierhuis, M. (1996). Selective ethnographic analysis: Qualitative modeling for work place ethnography. Paper presented at the Annual Meeting of the American Association of Anthropology, San Francisco, CA. Retrieved September 27, 2002, from AgentiSolution Web site: http://www.agentisolutions.com/documentation/papers/Aaa.pdf.

Sierhuis, M., and Clancey, W.J. (1997). Knowledge, practice, activities, and people. In B.R. Gaines and R. Uthurusamy (Eds.), *Artificial intelligence in knowledge management: Papers from the 1997 AAI Spring Symposium* (Technical Report SS-97-01) (pp. 142–148). Menlo Park, CA: American Association for Artificial Intelligence.

Sierhuis, M., Clancey, W.J., Sims, M.H. (2002). Multiagent modeling and simulation in human-robot mission operations work system design. In *Proceedings of the Hawaii International Conference on System Sciences* [CD/ROM] (10 pages). Computer Society Press. Los Alamitos, CA: IEEE Computer Society Press.

Sierhuis, M., Clancey, W.J., and van Hoof, R. (1999). *BRAHMS: A multiagent programming language for simulating work practice*. Retrieved September 25, 2002, from AgentiSolutions Web site: http://www.agentisolutions.com/documentation/papers/BrahmsWorkingPaper.pdf.

Sierhuis, M., Clancey, W.J., van Hoof, R., and de Hoog, R. (2000). Modeling and simulating human activity. In M. Freed (Ed.), *Simulating human agents: Papers from the 2000 Fall Symposium* (Technical Report FS-00-03) (pp. 100–109). Menlo Park, CA: American Association for Artificial Intelligence.

Sloman, A. (2001). Varieties of affect and the CogAff architectural scheme. From the *Symposium on Emotion, Cognition, and Affective Computing*, Society for the Study of Artificial Intelligence and Simulation of Behaviour (AISB). Brighton, England: University of Sussex.

Sloman, A. (2002, February). *The cognition and affect project. Exploring architectures for intelligent agents (whether natural or artificial)*. Retrieved February 13, 2003 from University of Birmingham School of Computer Science Web site: http://www.cs.bham.ac.uk/~axs/cogaff.html.

Sloman, A. (2003). How many separately evolved emotional beasties live within us? In R. Trappl, P. Petta, and S. Payr, (Eds.), *Emotions in humans and artifacts*. Cambridge, MA: The MIT Press.

Sloman, A., and Croucher, M. (1981). Why robots will have emotions. In *Proceedings on the 7th International Joint Conference on Artificial Intelligence* (pp. 197–202). Vancouver, BC.

Smith, B.R., and Tyler, S.W. (1997). The design and application of MIDAS: A constructive simulation for human-system analysis. Presented at the *2nd Simulation Technology & Training (SIMTECT) Conference*. Canberra, Australia.

Strohschneider, S. (2002). Automobiles, ind-col, and psychic systems: An essay on the functional perspective in cross-cultural psychology. In W.J. Lonner, D.L. Dinnel, S.A. Hayes, and D.N., Sattler (Eds.), *OnLine readings in psychology and culture*. Bellingham, WA: Western Washington University, Department of Psychology. Retrieved October 7, 2002, from Center for Cross-Cultural Research Web site: http://www.wwu.edu/~culture.

Suchman, L.A. (1987). *Plans and situated actions: The problem of human-machine communication*. Cambridge: Cambridge University Press.

Tambe, M. (1996). Teamwork in real-world, dynamic environments. In *Proceedings of the First International Conference on Multi-Agent Systems*. Menlo Park, CA: American Association for Artificial Intelligence.

Tambe, M., Adibi, J., Al-Onaizan, Y., Erdem, A., Kaminka, G.A., Marsella, S.C., and Muslea, I. (1999). Building agent teams using an explicit teamwork model and learning. *Artificial Intelligence*, *110*, 215–239.

Tenney, Y.J., and Spector, S.L. (2001). Comparisons of HBR models with human-in-the-loop performance in a simplified air traffic control simulation with and without HLA protocols: Task simulation, human data, and results. In *Proceedings of the 10th Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL: Simulation Interoperability Standards Organization.

Thibadeau, R., Just, M.A., and Carpenter, P.A. (1982). A model of the time course and content of reading. *Cognitive Science*, *6*, 157–203.

Turner, A., and Greene, E. (1977). *The construction and use of a propositional text base* (Technical Report No. 63). Boulder, CO: Institute for the Study of Intellectual Behavior, University of Colorado.

Tyler, S.W., Neukom, C., Logan, M., and Shively, J. (1998). The MIDAS human performance module. In *Proceedings of the Human Factors and Ergonomics Society 42nd Annual Meeting* (pp. 320–324). Santa Monica, CA: Human Factors and Ergonomics Society.

Warwick, W., McIlwaine, S., Hutton, R., and McDermott, P. (2001). Developing computational models of recognition-primed decision-making. *In Proceedings of the 10th Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL: Simulation Interoperability Standards Organization.

Wherry, R.J. (1976). The Human Operator Simulator – HOS. In T.B. Sheridan and G. Johannsen (Eds.), *Monitoring behavior and supervisory control* (pp. 283–293). New York, NY: Plenum Press.

Young, R.M. (1999). Brief introduction to ACT-R for Soarers: Soar and ACT-R still have much to learn from each other. Paper presented at *19th Soar Workshop*. Ann Arbor, MI: University of Michigan.

Yule, P., and Cooper, R. (2000, August). The COGENT tutorial. Presented at the *22nd Annual Conference of the Cognitive Science Society*. Philadelphia, PA.

Zacharias, G., and Baron, S. (1982). *A proposed crew-centered analysis methodology for fighter/attack missions* (BBN Technical Report 4866). Cambridge, MA: Bolt, Beranek, and Newman, Inc.

Zacharias, G., Baron, S., and Muralidharan, R. (1981). A supervisory control model of the AAA crew. In *Proceedings of the 17th Conference on Manual Control*, Los Angeles, CA.

Zacharias, G.L., Miao, A.X., Illgen, C., Yara, J.M., and Siouris, G. (1996). SAMPLE: Situation Awareness Model for Pilot-in-the-Loop Evaluation. In *Proceedings of the First Annual Symposium on Situational Awareness in the Tactical Air Environment*. Naval Air Warfare Center: Patuxent River, MD.

Zacharias, G.L., Miao, A.X., Kalkan, A., and Kao, S-P. (1994). Operator-based metric for nuclear operations automation assessment. In *Transactions of the Twenty-Second Water Reactor Safety Information Meeting* (NUREG/CP-0140, Vol. 1, pp. 181–205). Washington, DC: U.S. Nuclear Regulatory Commission.

Zachary, W.W., and Le Mentec, J-C. (2000). Modeling and simulating cooperation and teamwork. In *Proceedings of the Advanced Simulation Technologies Conference (ATSC 2000)*. San Diego, CA: The Society for Computer Simulation International.

Zachary, W., Campbell, G.E., Laughery, K.R., Glenn, F., and Cannon-Bowers, J.A. (2001). The application of human modeling technology to the design, evaluation, and operation of complex systems. In E. Salas (Ed.), *Advances in human performance and cognitive engineering research, Volume 1* (pp. 199–247). New York: JAI Press.

Zachary, W., Cannon-Bowers, J., Bilazarian, P., Drecker, D., Lardieri, P., and Burns, J. (1999). The Advanced Embedded Training System (AETS): An intelligent embedded tutoring system for tactical team training. *Journal of Artificial Intelligence in Education*, *10*, 257–277.

Zachary, W.W., Le Mentec, J-C., and Schremmer, S. (1996). GINA: A workbench for constructing cognitive agents. In the *Proceedings of the Human Factors and Ergonomics Society 40th Annual Meeting* (p. 864). Santa Monica, CA: Human Factors Society.

Zachary, W.W., Ryder, J.M., and Hicinbotham, J.H. (1998). Cognitive task analysis and modeling of decision-making in complex environments. In J. Cannon-Bowers and E. Salas (Eds.), *Decision-making under stress: Implications for training and simulation*. Washington, DC: American Psychological Association.

Zachary, W., Ryder, J., and Le Mentec, J-C. (2002). *Applied modeling of human competence and performance with COGNET and COGNET-P*. Unpublished manuscript.

Zachary, W., Ryder, J., Zubritzky, M., and Ross, L. (1990). *Application and validation of COGNET model of human-computer interaction in naval air ASW* (Technical Report 900430.8704). Spring House, PA: CHI Systems.

Zachary, W., Santarelli, T., Ryder, J., Stokes, J., and Scolaro, D. (2001). Developing a multi-tasking cognitive agent using the COGNET/iGEN integrative architecture. In *Proceedings of the 10th Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL: Simulation Interoperability Standards Organization.

Zhang, W., and Hill, R.W., Jr. (2000a). A template-based and pattern-driven approach to situation awareness and assessment in virtual humans. In *Proceedings of the Fourth International Conference on Autonomous Agents*. New York: Association for Computing Machinery (ACM), Special Interest Group for Artificial Intelligence (SIGART).

Zhang, W., and Hill, R.W., Jr. (2000b). Situation awareness and assessment—An integrated approach and applications. In *Proceedings of the 9th Conference on Computer Generated Forces and Behavioral Representation*. Orlando, FL: Simulation Interoperability Standards Organization.

Zubritsky, M., and Zachary, W. (1989). Constructing and applying cognitive models to mission management problems in air anti-submarine warfare. In *Proceedings of the 33rd Annual Meeting of the Human Factors Society* (pp. 129–134). Santa Monica, CA: Human Factors Society.

GLOSSARY

HBR Models Acronyms/Abbreviations:[25]

ACT	Atomic Components of Thought
AETS	Advanced Embedded Training System
APEX	Architecture for Procedure Execution
ART	Adaptive Resonance Theory
Brahms	Business Redesign Agent-Based Holistic Modeling System
CAPS	Concurrent Activation-Based Production System
CCT	Cognitive Complexity Theory
C-I Theory	Construction-Integration Theory
CogAff	Cognition and Affect Project
COGENT	Cognitive Objects within a Graphical EnviroNmentT
COGNET	Cognition as a Network of Tasks
DCOG	Distributed Cognition
EPIC	Executive Process/Interactive Control
HOS	Human Operator Simulator
Micro SAINT	Micro Systems Analysis of Integrated Network of Tasks
MIDAS	Man-machine Integrated Design and Analysis System
OMAR	Operator Model Architecture
PSI	See Footnote 18 on page II-79
SAMPLE	Situation Awareness Model for Pilot-in-the-Loop Evaluation
Soar	State, Operator, And Result

[25] The acronyms/abbreviations listed on page GL-1 are those of the 19 models discussed in this document.

2-D	two-dimensional
3-D	three-dimensional
A3I	Army Aircrew/Aircraft Integration
ACL	Allegro Common LISP
ACM	Association for Computing Machinery
ACTD	Advanced Concept Technology Demonstration
AD REM	Abstract Design Rendering Executable Models
ADS	Abstraction-Decomposition Space advanced distributed simulation
AFB	Air Force Base
AFRL	Air Force Research Laboratory
AI	artificial intelligence
AIAA	American Institute of Aeronautics and Astronautics
AISB	Artificial Intelligence and Simulation of Behaviour
ALSEP	Apollo Lunar Surface Experiments Package
AMBR	Agent-based Modeling and Behavior Representation
AMRL	Aerospace Medical Research Laboratory
ANN	Artificial Neural Net
ANSI	American National Standards Institute
AOEM	Air Operations Enterprise Model
APCHI	Asian Pacific Computer-Human Interaction Conference
API	application programming interface
ARC	Ames Research Center
ASTC	Advanced Simulation Technologies Conference
ASW	anti-submarine warfare
ATC	air traffic control
AWACS	Airborne Warning and Control System
BATON	Blackboard Architecture for Task-Oriented Networks
BBN	Bolt, Beranek, and Newman Inc.
BDI	Belief-Desire-Intention
BFTT	Battle Force Tactical Trainer
C2	command and control

CAD	computer-aided design
CART	Combat Automation Requirements Testbed
CC CAPS or 3CAPS	Concurrent, Capacity-Constrained Activation-Based Production System
CCBI	Center for Cognitive Brain Imaging
CCSIL	Command and Control Simulation Interface Language
CEL	COGNET Executable Language
CFOR	Command Forces
CGR	COGNET Graphical Representation
CHIL	Computer-Human Interaction Laboratory
CLIM	Common LISP Interface Manager
CLIPS	C Language Integrated Production System
CMN	Card-Moran-Newell
CMU	Carnegie Mellon University
CoABS	Control of Agent-Based Systems
CoLiDeS	Comprehension-based Linked model of Deliberate Search
CPM	Cognitive-Perceptual-Motor
CRA	Charles River Analytics, Inc.
CSIM	Crew/System Integration Model
CTA	cognitive task analysis
DARPA	Defense Advanced Projects Agency
DDD	Distributed Dynamic Decision-making
DERA	Defence Evaluation and Research Agency
DIS	Distributed Interactive Simulation
DMSO	Defense Modeling and Simulation Office
DoD	Department of Defense
D-OMAR	Distributed Operator Model Architecture
DOS	Disk Operating System
EC	Evolutionary Computation
FLAIRS	Florida AI Research Society
FLEX	Flavors Expert
FMARS	Flashline Mars Arctic Research Station

fMRI	functional magnetic resonance imaging
FORTRAN	Formula Translation/Translator
FTP	file transfer protocol
GDTA	Goal-Directed Task Analysis
GINA	Generator of Interface Agents
GLEAN	GOMS Language Evaluation and Analysis
GOMS	Goals, Operators, Methods, and Selection rules
GOMSL	GOMS Language
GPS	General Problem Solver
GRADE	Graphical Agent Development Environment
GUI	graphic user interface
HAIL	Human Automation Integration Laboratory
HAM	Human Associative Memory
HBR	human behavior representation
HCI	human-computer interaction
HLA	High-Level Architecture
HODAC	Human Operator Data Analyzer/Collator
I/O	input/output
IADS	Integrated Air Defense System
ICMAS	International Conference on Multi-Agent Systems
ICS	Institute of Cognitive Science
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IFAC	International Federation of Automatic Control
IFOR	Intelligent Forces
IJCAI	International Joint Conference on Artificial Intelligence
IMPRINT	Improved Performance Research Integration Tool
IPME	Integrated Performance Modeling Environment
IRL	Institute for Research on Learning
ISI	Information Science Institute
ISIS	Information Science Institute Synthetic
ISL	Integral Solutions Limited

IUSS	Integrated Unit Simulation System
KB Agent	Knowledge-Based Agent
LICAI	LInked model of Comprehension-based Action planning and Instruction
LISP	LISt Processing
LSA	Latent Semantic Analysis
LT	Logic Theorist
LTM	long-term memory
LTWM	long-term working memory
M&S	modeling and simulation
MA&D	Micro Analysis and Design
MCL	Macintosh Common LISP
MHP	Model Human Processor
MIL-AASPEM	Man-in-the-Loop Air-to-Air System Performance Evaluation Model
MIL-STD	Military Standard
MIT	Massachusetts Institute of Technology
ML	Markup Language
ModSAF	modular semi automated forces
MOPP	Mission Oriented Protection Posture
MRT	Multiple Resource Theory
MS	Microsoft
NASA	National Aeronautics and Space Administration
NGOMSL	Natural GOMS Language
NL	natural language
NRC	National Research Council
NT	New Technology (Microsoft Windows operating system)
OCM	Optimal Control Model
ONR	Office of Naval Research
OPL	Operator Procedure Language
OS	operating system
OTS	off-the-shelf

PARC	Palo Alto Research Center
PC	personal computer
PDL	Procedure Definition Language
PEO STRI	Program Executive Office for Simulation, Training, and Instrumentation
PET	positron emission tomography
PI	Principal Investigator
PPS	Parsimonius Production System
PROCRU	Procedure-Oriented Crew Model
PSA	Personal Satellite Assistant
R&D	research and development
RAP	Reactive Action Package
RL	resolution level
RPD	Recognition-Primed Decision-making
RTI	Run-Time Interface
SAE	Society of Automotive Engineers
SAINT	Systems Analysis of Integrated Network of Tasks
S-CAPS	situated CAPS
SCORE	Simulation Core
SDK	Software Development Kit
SFL	Simple Frame Language
SIGART	Special Interest Group for Artificial Intelligence
SIGCHI	Special Interest Group on Computer-Human Interaction
SIMTECT	Simulation Technology and Training
SL	selection threshold
SMC	Systems, Man, and Cybernetics Society
SME	subject matter expert
STEAM	Shell for TEAMwork
STM	short-term memory
STOW-97	Synthetic Theater of War-1997
STOW-E	synthetic Theater of War-Europe

STRICOM	U. S. Army's Simulation, Training, and Instrumentation Command
TCP/IP	Transmission Control Protocol/Internet Protocol
TCV	terminally configured vehicle
TEX	Task Experimenter
TM	Technical Memorandum
TR	Technical Report
UAV	unmanned aerial vehicle
UIST	User Interface Software and Technology
UMTRI	University of Michigan Transportation Research Institute
USC	University of Southern California
VTOL	vertical take-off and landing
WM	working memory
XML	eXtensible Markup Language

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED *(From–To)*
August 2003	Final	February 2002–June 2003

4. TITLE AND SUBTITLE

A Review of Computer-Based Human Behavior Representations and Their Relation to Military Simulations

5a. CONTRACT NUMBER
DAS W01 98 C 0067/DAS W01 02 0012

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

John E. Morrison

5d. PROJECT NUMBER

5e. TASK NUMBER
AK-2-2190

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Institute for Defense Analyses
4850 Mark Center Drive
Alexandria, VA 22311-1882

8. PERFORMING ORGANIZATION REPORT NUMBER

IDA Paper P-3845

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

DMSO
1901 N. Beauregard Street
Suite 500
Alexandria, VA 22311-1705

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution unlimited. (18 August 2004)

13. SUPPLEMENTARY NOTES

14. ABSTRACT

One of the goals of the Defense Modeling and Simulation Office (DMSO) has been to promote the development and assessment of computational human behavior representations (HBRs) that potentially provide synthetic forces—both Red and Blue—for live, virtual, and constructive military simulations. This paper reviews the domain of HBRs that could be integrated with military simulations. The intent is to provide the modeling and simulation (M&S) community an understanding of specific HBR models and to identify specific interoperability problems. The study identified 19 different HBRs that have at least some applicability to military simulations. Analyses of these models suggested the following generalizations concerning the current state of the art in human behavior modeling: (1) Decision-making is a universal function of all models; (2) All models can represent some form of memory storage and retrieval functions; (3) Both the "front-end" of cognition (perception and attention) and cognitive output (psychomotor action) are represented in most models; (4) Because most models do not include learning functions, they may not react appropriately to novel situations; (5) The capability to emulate situation awareness (SA) is explicitly represented in only a few models; and (6) Very few of the models have the capability to simulate emotional or social behaviors.

15. SUBJECT TERMS

cognitive model, human behavior representation (HBR), modeling, simulation, training

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
					Col. Michael L. Finnern
a. REPORT Uncl.	b. ABSTRACT Uncl.	c. THIS PAGE Uncl.	SAR	152	19b. TELEPHONE NUMBER *(include area code)* (703)998-0660

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18